UNIVERSITY OF LONDON THESIS

Degree PhD Year 2008 Name of Author LAI, Zhaohong

Department of Electronic and Electrical Engineering
University College London
University of London

Towards Automatic Traffic Classification and Estimation of Available Bandwidth in IP Networks

Zhaohong Lai

A thesis submitted in partial fulfilment of the requirements for the degree of Doctor of Philosophy in the Faculty of Engineering of the University of London, and for the Diploma of the University College London

March 2008

ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346

Declaration

I, Zhaohong Lai, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the thesis.

Signature:

Print Name: **Zhaohong Lai**

Abstract

Growing rapidly, today's Internet is becoming more difficult to manage. A good understanding of what kind of network traffic classes are consuming network resource as well as how much network resource is available is important for many management tasks like QoS provisioning and traffic engineering. In the light of these objectives, two measurement mechanisms have been explored in this thesis.

This thesis explores a new type of traffic classification scheme with automatic and accurate identification capability. First of all, the novel concept of IP flow profile, a unique identifier to the associated traffic class, has been proposed and the relevant model using five IP header based contexts has been presented. Then, this thesis shows that the key statistical features of each context, in the IP flow profile, follows a Gaussian distribution and explores how to use Kohonen Neural Network (KNN) for the purpose of automatically producing IP flow profile map. In order to improve the classification accuracy, this thesis investigates and evaluates the use of PCA for feature selection, which enables the produced patterns to be as tight as possible since tight patterns lead to less overlaps among patterns. In addition, the use of Linear Discriminant Analysis and alternative KNN maps has been investigated as to deal with the overlap issue between produced patterns. The entirety of this process represents a novel addition to the quest for automatic traffic classification in IP networks.

This thesis also develops a fast available bandwidth measurement scheme. It firstly addresses the dynamic problem for the one way delay (OWD) trend detection. To deal with this issue, a novel model - *asymptotic* OWD *Comparison* (AOC) model for the OWD trend detection has been proposed. Then, three statistical metrics SOT (Sum of Trend), PTC (Positive Trend Checking) and CTC (Complete Trend Comparison) have been proposed to develop the AOC algorithms. To validate the proposed AOC model, an avail-bw estimation tool called *Pathpair* has been developed and evaluated in the Planetlab environment.

Acknowledgement

Foremost, my thanks go to Prof. Chris Todd. I am honored to have Prof.Todd as my supervisor and sincerely thank him for his invaluable guidance and consistent support for my Ph.D study. Without his encouragements and support, this thesis would not have been possible. I especially thank Dr Miguel Rio for his inspiration and thoughts to improve the thesis. He has been actively involving in my research and has been always available to advise my work. Prof. Alex Galis who enabled me to work in the two EU projects (mCDN and AN Phase2) has fully supported my work, which I truly appreciate. I would also like to express my gratitude to Prof. II Song for providing me many constructive suggestions on the thesis. My thanks also go to Prof. Izzat Darwazeh for his support and valuable comments on my work. I would also like to thank Prof. Mark Handley for assisting me in the experimental work.

During my full-time PHD study at UCL, I was lucky to work in a stimulating and friendly research environment - UCL E&EE Networks and Services Research Laboratory, and grateful to the members who have provided me the substantial support on my work. I would also like to thank the members of the mCDN and AN team for sharing the research experience, specially to Daniel Pakkala(VTT), Yannis Matalas(IntraCom), Tiziana Toniatti (Siemes Mobile), Joachim Sokol(SAG), Klaus Peter(Fokus), Nikos Dragious (NTUA), Meng Song and Bertrand Mathieu(France Telecom).

I would like to extend my gratitude to the Secretary of State for Education and Science as my PHD study was supported by the Overseas Research Students Award Scheme (2004-2007).

Finally, and most of all, I would like to thank my family for their moral encouragements and support: my parents, brothers, nieces and dearest JinJin; this thesis is dedicated to you.

Contents

List of Figures

List of Abbreviations

ABW	Available Bandwidth
ABWE	Available Bandwidth Estimation
ADR	Asymptotic Dispersion Rate
AIMD	Additive Increase Multiplicative Decrease
AN	Ambient Network
ANN	Artificial Neural Network
AOC	Asymptotic One way delay Comparison
AOWD	Asymptotic One Way Delay
ATA	Available bandwidth Turning Area
ATC	Automated Traffic Classifier
ATM	Asynchronous Transfer Mode
ATP	Available bandwidth Turning Point
BART	Bandwidth Available in Real-Time
BLINC	BLINd Classification
CDN	Content Distribution Network

CM	Capacity Mode
CORS	Context-Aware Optimal Route Selection
COS	Class of Service
CPU	Central Process Unit
CTC	Complete Trend Comparison
DFR	Decreasing Failure Rate
DEM	Double End Model
DiffServ	Differentiated Services
DARPA	Defence Advanced Research Project Agency
DSCP	Differentiated Services Code Point
ECMP	Equal Cost Multi Path
EDS	EDge Server
FCBF	Fast Correlation-Based Filter
FCFS	First Come First Serve
HTTP	Hyper Text Transmission Protocol
IANA	Internet Assigned Numbers
IC	Interrupts Coalescence

ICMP	Internet Control Message Protocol
IGI	Initial Gap Increasing
IMG	Internet Media Guide
IPDR	IP Detailed Record
IRC	Internet Relay Chat
ISP	Internet Service Provider
IntServ	Integrated Services
KDE	Kernel Density Estimation
KNN	Kohonen Neural Network
KNNA	Kohonen Neural Network Analyser
LAN	Local Area Network
LDA	Linear Discriminant Analysis
LLF	Long-lived Flow
LRD	Long-Range Dependence
LSP	Label Switched Path
mCDN	multimedia Content Discovery and Delivery Network
ML	Machine Learning

MPEG	Motion Picture Experts Group
MPLS	Multi Protocol Label Switching
MRTG	Multi-Router Traffic Grapher
MTU	Maximum Transmission Unit
NAPT	Network Address Port Translation
NCE	Network Capacity Estimation
NCP	Network Control Protocol
NIC	Network Interface Card
OC	Optical Carrier
OWL	Web Ontology Language
OWL-DL	Web Ontology Language Description Logics
ORS	Optimal Route Service
OS	Operation System
OSPF	Open Shortest Path First
OWD	One Way Delay
P2P	Peer to Peer
PC	Principal Component

PCA	Principal Component Analysis
PCAP	Packet Capture
PCT	Pairwise Comparison Test
PDSM	Performance Degradation at Sender Machine
PDT	Pairwise Difference Test
PGM	Probe Gap Model
PNCM	Post-Narrow Capacity Mode
POP3	Post Office Protocol 3
PRM	Probe Rate Model
PS	Processor Sharing
PTC	Positive Trend Checking
PTR	Packet Transmission Rate
QoS	Quality of Service
QOSPF	QoS-enabled Open Shortest Path First
RTP	Real-time Transport Protocol
RTT	Round Trip Time
SATO	Service-aware Adaptive Transport Overlay

SCDR	Sub-Capacity Dispersion Range
SEM	Single End Model
SLA	Service Level Agreements
SLF	Short-lived Flow
SMTP	Simple Mail Transfer Protocol
SNMP	Simple Network Management Protocol
SLoPS	Self-Loading Periodic Streams
SOT	Sum of Trend
SPF	Shortest Path First
STP	Signalling Transfer Point
SS7	Signalling System No.7
TCP	Transmission Control Protocol
TOPP	Train of Packet Pair
UDP	User Datagram Protocol
VoIP	Voice over IP
WAN	Wide Area Network
WWW	World Wild Web

Chapter 1

Introduction

1.1 Background

The Internet, 'the network of networks', is one of the most successful stories in modern technologies and exerts deeper and broader impact on our daily life. In the past decade, as the Internet has been expanded dramatically, ISPs and local network managers are facing the increasing challenge to collect network feedback. This thesis focuses on the study of collecting two different types of network feedback: one is what kind of network traffic is consuming network resource while the other is how much available bandwidth is left for a given network path. Both of them are important in many network management tasks such as QoS provisioning, traffic engineering and security filtering etc as discussed below.

On the one hand, as network users are allowed to inject their own network applications (which were developed according to their needs) into the Internet, recent years have seen the dramatic increase in the number and variety of network services from video conference to content delivery, to remote education and to online shopping etc. In the earlier days of the Internet, network applications were bound to well-known or registered port numbers, it was easy and straightforward to classify them. However, many recent private network applications use unregistered ports and therefore they are unseen to network operators. In particular, it has been reported that some network applications are deliberately designed to bypass network firewalls so as to gain their own benefits. Some of them maybe are used to attack networks; others may be used for relaying traffic (i.e. Skype) [BMMR07]. Therefore, for network operators, understanding the nature of network applications is an essential task as they need to ensure that those network applications conform to their policies and regulations and exert no harm to networks.

On the other hand, the fast expansion in Internet gives rise to challenges for ISPs in terms of network bandwidth measurement. The traditional network bandwidth measurement approach is implemented in a passive way, a scheme that is realised by passively collecting data from discrete nodes. However, this approach is becoming much less effective in the IP network environment because of the large scale nature of the Internet, not to mention that its topology is changing from time to time. In addition, as different ISPs implement varying policies, some of nodes may fail to provide the required data for the passive collection approach. Another more promising approach is to estimate network bandwidth through the active probing approach. This approach is not subject to the boundaries that the passive approach has because it only involves two end nodes of a detected path. Hence, it is important to have an active way for network bandwidth estimation in the current Internet environment.

In practice, it is often the case that both traffic classification and bandwidth estimation functionalities are required for a network management task, as illustrated in the following scenario.

1.2 A Scenario: Traffic Classification and Bandwidth Estimation used in multimedia Content Discovery and Delivery Networks (mCDNs)

With the explosive growth in web-based content, Content Distribution Networks (CDNs) are gaining popularity by improving user access latency, throughput, reliability and scalability [JM04]. In order to push content closer to clients, distributed Edge Servers (EDS) are deployed in different areas/countries [MOL04] in a CDN system. As each EDS is a core node, the high-speed links are deployed within the EDS network domains and backup links are often installed. As shown in Fig 1.1, multiple paths often co-exist between two ends so as to guarantee the connection reliability of an EDS.

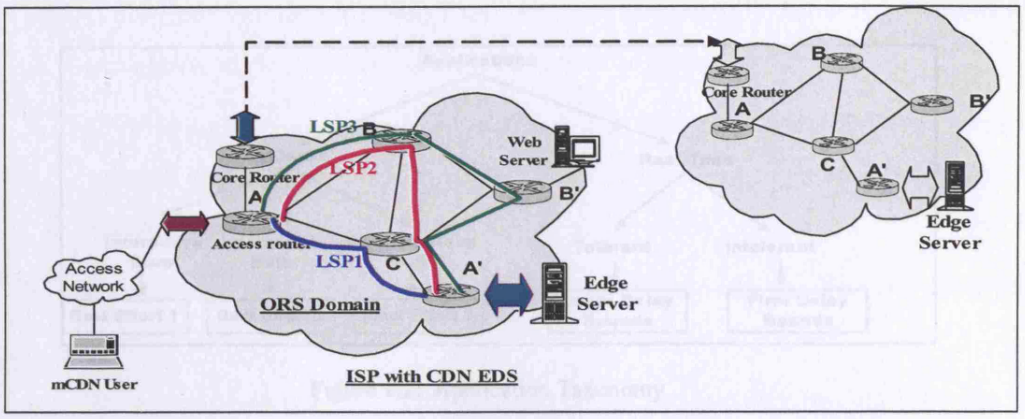


Figure 1.1: Multiple paths coexist between two ends (AA')

To effectively use those multiple network paths, a component called Context-Aware Optimal Route Service (CORS) has been developed [LAI05a][LAI05b][LAI05c] in the mCDN project. The optimisation decisions of CORS are made according to two types of contextual information. One is network related information like bandwidth and network topology. It is the kind of information that says how much network resource can be provided. The other is content and application related information. This is the kind of information that says how much resource a piece of content is requesting. In CORS, the former is realised by a Context Sensor [LAI05a] to collect required bandwidth data and the latter is implemented by Internet Media Guide (IMG), a component used to describe content features with relevant metadata [LAI05b]. Note that another effective alternative to provide content related information is through traffic classification. The study of [SDZ94] illustrated such an example. In [SDZ94], Shenkef et al proposed a taxonomy regarding how to map network application to relevant network services as shown in Fig 1.2. Although Fig 1.2 only shows two network classes (elastic or real time), the latest studies like [MP05][MZ05] are already able to classify as many as ten traffic categories.

This scenario shows that how the two measurement schemes can be put together to help build the optimisation functionalities. Aside from that, the two schemes also play important roles in many areas such as QoS provisioning, traffic engineering and TCP congestion control etc. The next section will detail the specific objectives of this research.

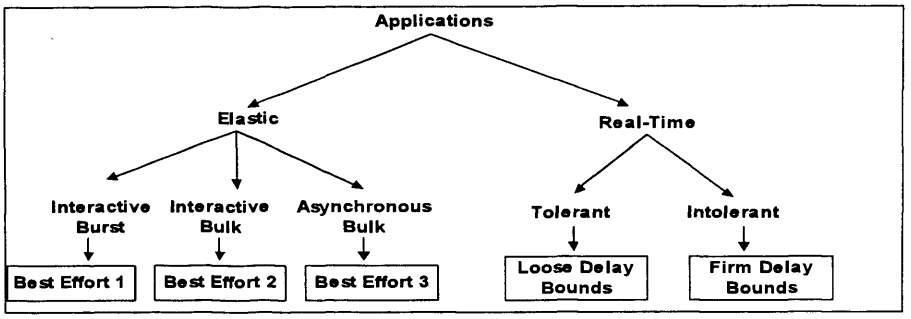


Figure 1.2: Application Taxonomy

1.3 Aims and Objectives

In the light of the motivation as discussed above, this thesis aims to explore two measurement mechanisms in IP networks: one is for available bandwidth estimation while the other is for traffic classification.

Objectives in Traffic Classification

This thesis aims to develop a traffic classification scheme that is capable of detecting network traffic automatically. The term 'automatic' is defined in Oxford Dictionary as '*operating with little or no direct human control*' [OD01]. In this thesis, the goal of developing an automatic traffic classifier is to minimise the human involvements that traffic classification requires. The motivation behind this is because one of the challenging issues for existing traffic classification schemes is that prior traffic analysis is an essential requirement for them to classify unknown traffic [MP05]. Apparently, the prior traffic analysis based scheme is infeasible nowadays as the number of new applications is growing so fast [MZ05].

In this thesis, the following objectives will be addressed in order to fulfil the automatic classification purpose.

- To define and develop concepts that help understand features of network categories and inputs that can be used for traffic classification.
- To develop an unsupervised self-organising scheme that is able to automatically produce traffic patterns in a low-dimensional map from high-dimensional input space.
- To develop a scheme that is capable of performing the feature selection functionalities so as to produce traffic patterns as tight as possible.
- To evaluate the overlap issues between traffic patterns and develop schemes to tackle the overlap problem.

Objectives in Available Bandwidth Estimation

The key goal of the research in avail-bw estimation is how to improve the measurement speed while also preserving comparable accuracy in comparison with current major tools. As will be discussed in Chapter 5, there exist a number of avail-bw estimation tools that are realised with the active probing way. Of these tools, it has been reported [SMH05] that *Pathload* which is based on the one way delay (OWD) trend detection is one of the most accurate tools. This research extends *Pathload's* work and aims to develop a faster tool. The research objectives include the following aspects:

- To investigate the problems in existing OWD based avail-bw tools.
- To define a new model to enhance the reliability of the OWD trend detection as well as the trend detection speed.
- To develop algorithms for the proposed avail-bw estimation model.
- To develop a new avail-bw tool based on the proposed model and validate its performance and accuracy in a real network environment.

1.4 The Thesis Structure

In accordance with the above objectives, the rest of this thesis is organised as follows.

Chapter 2 introduces the basic definitions for network applications and traffic categories and discusses the motivations of traffic classification. In addition, it presents a high-level overview of prior studies in traffic classification and addresses the issues remaining in each classification approach. Also, a comparison of the proposed scheme with current major work has been given in this chapter.

Chapter 3 proposes an IP flow profile based automatic traffic classification scheme, which is realised through the use of the KNN technique, a technique that is able to automatically produce different traffic patterns in a KNN feature map as its learning algorithms which are Gaussian-based. First of all, this chapter discusses the use of the context concept to describe input data as well as the use of IP flow profile to organise rich context inputs at flow level. Then it goes further to model IP flow profile according to IP header based five context inputs. After that, this chapter presents the ontological view of the relationship between the proposed classification concept and other concepts. Furthermore, it investigates how to reduce classification cost through selecting long-lived flows in real networks. Next, it examines why and how Kohonen Neural Network can be used for classification and is followed by the implementations of a series of modules that used for classification. Subsequently, it presents a number of proof-of-concept experiments to validate the automatic features of the proposed classification scheme.

Chapter 4 addresses the accuracy issue in traffic classification. It firstly discusses the feature selection issue and how to deal with that with the Principal Component Analysis (PCA) technique. Afterwards, it goes on to address the overlap problem and proposes the use of Linear Discriminant Analysis (LDA) and alternative maps deal with overlaps to achieve high accuracy detection results. Then, it presents a number of experiments to illustrate the feature selection results with PCA and the classification results after using LDA and alternative maps. The classification performance particularly the speed of

classification has been addressed at the end of this chapter. The following figure shows the relationship between chapters 3 and 4.

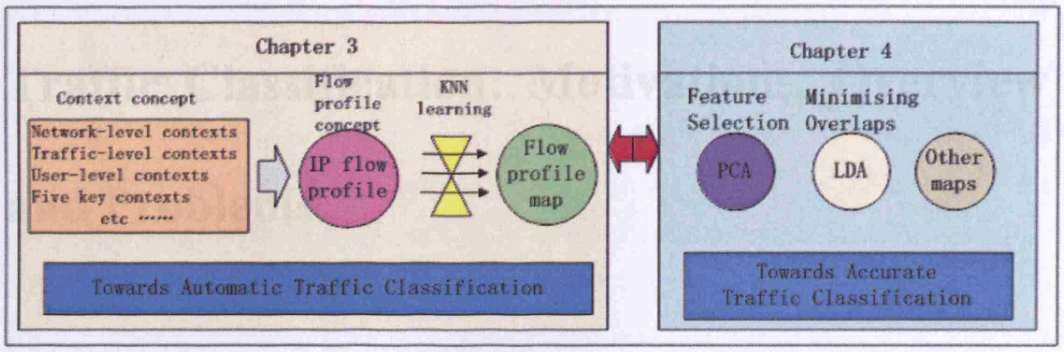


Figure 1.3: Towards Automatic and Accurate traffic classification

Chapter 5 starts with an introduction to the basic definitions of bandwidth, network capacity and available bandwidth concepts. Then, it presents the underlying rationales including the bottleneck spacing effect and packet dispersion for active bandwidth estimation and is followed by detailed analysis of how packet dispersions would be affected by cross traffic. In this thesis, the cross traffic also called competing traffic means the traffic that does not belong to the probing traffic. After that, it discusses whether packet dispersions can be used for avail-bw estimation and compares two avail-bw estimation models: the Probe Gap Model and the Probe Rate Model. Subsequently, it reviews the current major avail-bw estimation tools and gives relevant comparisons between them.

Chapter 6 develops a fast avail-bw tool called *Pathpair* based on the OWD trend. Firstly, it presents the motivations behind the work. Then, it proposes a novel solution in the form of an asymptotic OWD comparison model for OWD trend detection and presents a detection algorithm through the use of three metrics: A_{SOT}, A_{PTC} and A_{CTC}. After that, it addresses the design issues of *Pathpair* and gives the implementation architecture of *Pathpair*. Next, it presents a number of experiments to validate the proposed asymptotic OWD comparison model as well as *Pathpair*'s accuracy and performance.

Chapter 7 summarises this thesis and presents some future work.

Chapter 2

Traffic Classification: Motivations, Overview and Problems

2.1 Introduction

Classification of network traffic plays important roles for many network management tasks such as traffic engineering [MIK05], service class mapping [RSS04] and security filtering [ZAG06] etc. Classifying network traffic is proving challenging [JMSW07][JEF07]. In the early Internet, traffic classification could be easily realised by reading port numbers. With the rapid evolution of network services, many applications are becoming more difficult to classify as they can be formed in various ways like using unregistered ports or wrapping in other protocols (i.e. HTTP). Currently, there are a number of tools that are able to classify different types of traffic. Some of them are able to detect P2P traffic effectively [MC06] and others are dedicated to classifying *'chatting'* traffic etc [DWF03]. They are realised with varying techniques such as payload decoding, signature scanning, traffic statistical feature analysis etc. To address the difference between different classification techniques, this chapter presents a high-level review of the current major solutions for traffic classification and analyse related pros and cons in different classification approaches. It is structured as follows. Section 2 defines the concepts of network application and traffic category in the context of this thesis. Section 3 discusses the motivations behind this work. Section 4 presents the up-to-date review of the major approaches and algorithms on traffic classification appearing in the last few years. In addition to that, the outstanding issues and problems in each classification approach have been fully discussed and analysed. Section 5 gives a brief comparison of the proposed classifier with existing work. The conclusion is given in the last section.

2.2 Concepts: Network applications and Traffic Categories

As defined in [OD01], *an application* refers to a computer program designed to fulfill a particular purpose. Similarly, a network application can be understood as a network program to accomplish a specific task or service. The concept of traffic category usually has got a broader scope than that of network application. A traffic category often refers to a group of network applications with common features like traffic profile, similar service requirements [MP05]. Hence, a traffic category consists of at least one application. An identical term to traffic category is traffic class. However, it should be noted that a network application may not only belong to one traffic category. For example, an http-based application has the potential to carry a number of traffic categories like Games and Multimedia. One question therefore is raised about how many traffic categories exist in current IP networks. There is not a definitive answer yet and this question depends on the purpose of using traffic classification. In the context of security protection, being able to classify the Malicious class from Normal traffic class is normally enough to satisfy corresponding requirements [MZ05]. [RSS04] partitions traffic into four classes as it aims to map traffic into different Classes of Service (CoS) according to varying traffic classes. The four defined classes are: Interactive, BULK, Streaming and Transaction [RSS04]. [MZ05] categorises network traffic into nine classes according to the potential requirements of network infrastructure. This thesis shares the same view as [MZ05] which is developed under the background that traffic classification can be used for a variety of network services such as service class mapping, optimal route service and optimal traffic rerouting in overlay networks etc. as discussed below.

2.3 Motivations

Classification for Active QoS approach

The best-effort-only network architecture was ideal for traditional network applications i.e. e-mail, telnet and ftp, as their requirements are rather elastic in terms of network performance [MIK05]. Recently, due to the fast growing number and variety in network applications, the ability to support various service classes is becoming an important issue for network operators. In particular, as many of them are time and bandwidth sensitive or mission-critical applications, recent years have seen a great deal of effort to construct QoS-enabled network infrastructure including schemes such as DiffServ [RFC2475], IntServ [RFC1633], and QoS-based routing [WC96][SRS97][JNG04] etc. Different service classes usually are defined in ToS (Type of Service) [RFC791] or DSCP (Differentiated Services Code Point) [RFC2474]. The implementations to categorise network traffic into different service classes mainly rely on the Service Level Agreements (SLA) [CIS05][JMSW07]. In practice, such implementations are infeasible in a large scale network as policies and agreements vary largely in different regions and countries. One alternative to realise the service class marking is through traffic classification. Unlike an SLA that involves a great many policies and manual agreements, traffic classification to be deployed at any part of ISPs' networks, allowing the service class marking to be realised in an active and independent way. However, traffic classification is still under research. As stated in [RSS04], one of the key factors holding back widespread QoS implementation is the absence of traffic classification.

Network Application Legitimacy Issues

IP networks are open to any form of traffic which can be from either registered or unregistered applications like ftp or unknown P2P services. In such an environment local network managers desperately need to know whether the running applications are legitimate and in accordance with their policies. The failure to identify such applications is a threat to many organisations as undesired network usage can seriously degrade the network performance and available bandwidth [SUH06]. There are at least two reasons. Firstly, many P2P based applications use machines that have good connections as relay nodes for other clients [BAS06][LJ06]. Secondly, many new emerging services like Skype have a degree of stealth and are far beyond the detection ability of existing detection tools.

Hence, it is important to have a traffic classification scheme to tackle such type of legitimate issues.

Classification in Network Optimisations

ISPs can also use traffic classification to optimise network usage, pricing and routing [MC06][LAI07]. As stated in [MIK05], traffic behaving in a particular manner should be classified in the same class as other traffic behaving in a similar manner according to the pre-specified policies. Such policies are defined by ISPs according to the requirements of traffic management. For example, for the purpose of optimising network resource, ISPs can categorise the network traffic into P2P traffic and non-P2P traffic and during the busy day time (i.e.9:00am-11:00am), ISPs can deliberately throttle P2P traffic so as to allocate more network resource to non-P2P traffic. Besides, ISPs can assign different priority schemes or policies to the various traffic classes to maximise the total revenue, according to relevant business models [MIK05].

Classification in helping build SATO for Ambient Networks

The aim of Ambient Network (AN) project [EP07][NIE04] is to foster co-operation between the next generation heterogeneous wireless networks, in order to gather resources within and across ANs to provide new services [GAL06]. One innovative design in the Ambient Network is to provide service-aware transport functionalities, which are implemented within a set of service-specific overlays, which are called Service-aware Adaptive Transport Overlays (SATOs). To implement SATO functionalities, one of the requirements concerns the interactions between overlay and underlay layers. For example, it is important to know what kind of services are running in ANs. Traffic category information is very useful to address this issue. Specifically, such a task can be realised by mapping service classes according to traffic categories as stated in [RSS04]. Then,

according to different service classes, we can achieve the service adaptation functionality. For instance (Fig 2.1), when congestion happens on one overlay path like path 1 from *Onode(Overlay node)* 1 to *Onode* 4, the time-sensitive traffic detected by sensors will be rerouted to other better paths i.e. path 2. Currently, it is proposed to deploy traffic classification sensors into super peers of each ambient network. After deriving service classes according to information from sensors, service adaptation tasks can be developed.

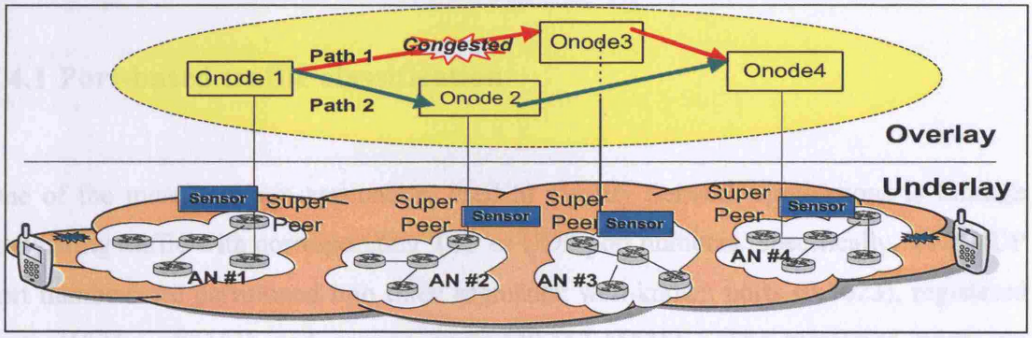


Figure 2.1: A simplified overlay and underlay network structure in ANs

2.4 Related work

Recently, a variety of studies on traffic classification have appeared. The approaches used for traffic classification can be divided into the following four classes according to which measurement parameters are being used:

- *Port-based classification* simply uses transport layer TCP or UDP port numbers as the input parameters for traffic classification.
- *Payload-based classification* uses parameters derived from either the payload decoding or signature scanning for traffic classification.
- *Transport-layer based classification* uses information mainly based on network and transport layer for classification.

- *Statistics based classification* uses a large number of parameters obtained from the analysis of network traffic features and flow characteristics.

This section presents a high-level overview of the above four classification approaches, and highlights the problems and issues that remain.

2.4.1 Port-based traffic classification

One of the most common approaches used to identify network applications is through associating traffic with corresponding TCP or UDP port numbers. Specifically, TCP/UDP port numbers are partitioned into three segments: well-known ports (0-1023), registered ports (1024 - 49,151) and private ports (49,152-65535). The registered ports are managed by the organization called IANA (Internet Assigned Numbers) [IANA]. It is clear that network users can develop their own programs using private ports, which will reduce classification accuracy accordingly. Apart from this issue, there also exist quite a few outstanding problems.

1. There is a dramatic increase in the number of network applications that are wrapped into well-known ports [MZ05], for the purpose that these applications can then pass through the security filtering or local policies that are defined in enterprise or campus networks. For example, Skype often uses HTTP-based port numbers as its data transmission ports. In principle, as stated in [RSS04], the available implementations of IP over HTTP allow the tunneling of all applications through port 80 or 8080.

2. In principle, the registered ports should be only used by pre-specified applications. But nowadays they have been often used by the private applications. This certainly will cause misclassification of the traffic using these registered ports but carrying private applications.

3. The use of widespread NAPT (Network Address Port Translation) enables network clients to escape from port-based traffic classification [WZA06].

4. As stated in [ZNA05], non-privileged users often have to use ports above 1023 to deliver those applications that should use well-known ports.

It is clear from the above, port-based classification solutions are unable to produce accurate results. For example, it has been reported that the port-based classification can correctly identify only 50%-70% of the total traffic [MP05]. As the number of applications using unregistered ports or tunnelled into well-known ports is rising [MZ05], the port-based classification is becoming less reliable.

2.4.2 Payload based classification

To overcome the port-based classification issues, the payload based approach has attracted a great deal of attention [HSS05][DWF03][MP05][KPF05]. Realisation of this approach requires the deep analysis of packet payload to see whether distinct signatures exist for known applications. The term signature, as defined in [HSS05], refers to a set of conditions defined over a set of features in the application traffic. [DWF03] uses signatures to identify different chat applications like IRC (Internet Relay Chat) [OR03], web-based chatting applications etc. [SSW04] presents an efficient approach i.e. the packet-level trace to identify P2P traffic with predefined signatures. [MP05] integrates nine detection techniques together while six of them belong to the payload-based classifications. Although this approach is complicated, high accuracy (over 90%) has been achieved. In addition, since the [MP05] work is based on the manual signature analysis, as pointed out in [HSS05], it involves a labor-intensive process to develop potential signatures and requires multiple iterations to improve the accuracy. Hence, [HSS05] explores an automated signature construction scheme to detect network applications using

statistical machine learning algorithms. However, it is still unable to yield signatures from encrypted content.

In summary, a number of issues still remain for the port-based classification schemes:

1. First of all, the main complication is that prior knowledge of the detected application's protocol signatures, protocol interactions and packet format is a must in payload-based classification techniques [KPF05]. If related signatures are not available, other mechanisms should be considered rather than the payload-based approach.

2. This approach requires considerable processing and memory for deducing signatures and it requires collecting and analysing multiple complete datagrams eg it requires collecting the first K bytes as well as concurrent analysis of a potentially large number of flows [ZNA05][DBL03].

3. It is a challenging task to have up-to-date and complete signatures for all the traffic for they may change over time.

4. Signatures are often not open in commercial applications.

5. Many applications like BitTorrent deliberately use obfuscation methods eg variable-length padding so as to elude payload-based classification [EAM06].

6. The approach of this type is unable to operate on the encrypted traffic as encryption will render the data content unintelligible [DBL03].

7. The loss of UDP packets that contains required signature information will also result in the failure of this approach.

8. Finally, as stated in [ZNA05], as this approach requires direct access and analysis of application layer content, it imposes an explicit breach of organisational privacy policies or violation of relevant privacy legislation.

2.4.3 Transport-layer based classification

The transport-layer based approach is a scheme realised through the analysis of network connections when network applications communicate with networks. This scheme is often used in P2P traffic identification as P2P applications usually possess larger numbers of network connections than ordinary applications. [WS04] develops a traffic scheme to classify P2P traffic according to the IP address distribution. [MC06] classifies P2P traffic based on the analysis of network diameter[1], a similar concept to the IP address distribution. [KPF05] introduces a new mechanism called BLINC (*BLINd Classification*) for traffic classification according to transport-layer information. BLINC associates Internet host behaviors and patterns with applications by examining three levels: social, functional and application levels. Although BLINC is able to accurately associate hosts with the services they provide or use, it can not classify a single TCP flow [BTA06], i.e. in the case where there is only one connection. One of the major problems of this approach is that it is impossible to get all the required hosts information if the traffic classification is performed in the network core.

2.4.4 Statistics-based traffic classification

Another approach to traffic classification is based on the statistical feature analysis of network traffic. It recently has received a great deal of attention [DWF03][DBL03] [MZ05][ZM05][MHL04][EAM06][RSS04][BTA06], as detailed below.

Though not strictly for traffic classification, earlier work attempts to develop mathematical models to describe various types of network traffic. Leland et al claimed that LAN traffic follows the self-similar distribution [LEL94][KMF04a][KMF04b]. Paxson et al also found the self-similar distribution in WAN traffic in [PAX95]. Similarly,

[1] Given two nodes, the associated network diameter means the hop number between them. More details can be seen in [MC06].

Crovella showed the WWW[2] traffic followed the self-similar distribution as well [CB97]. [PAX95] stated that a Poisson process could describe a number of human-initiated events like ftp and telnet connection packets. Nevertheless, [PAX95] shows that the burst traffic like FTPDATA has heavy-tailed features. For example, the upper 5% tail per FTPDATA burst fits well to a Pareto distribution with parameter $0.9 \leq \alpha \leq 1.4$ [PAX94]. Pareto also applies to the distribution of WWW size [DOW01a][DOW01b].

Parish et al develop an application detector according to the features of the packet size distribution [DBL03]. The relevant results show that the classifier often requires 30 seconds and needs about 10% of the whole streams. In addition, Parish et al observe that the change of the packet size distribution is not significant even after traffic has been encrypted [DBL03].

- Supervised

In [MP05] and [MZ05], Moore and Zuev present a supervised Naïve Bayesian classifier[3] based on the assumption that each discriminator for classification follows the Gaussian distribution. The Kernel Density Estimation technique is used to describe the discriminators when discriminators are not exactly Gaussian distributed. The Fast Correlation-Based Filter (FCBF), a technique realised through the entropy measurement on variables, is used for the fast feature selection. Without any refinements, the classification accuracy only reaches about 65% while it has been raised to 93.5% and 94.29% after applying the Kernel Estimation and FCBF techniques respectively [MZ05]. However, as Jiang et al stated, this classifier is complex with high cost on the data collection, aggregation and generation [JMSW07]. In [AM07], a traffic classification based on Bayesian Neural Networks has been reported.

[2] World Wide Web
[3] A machine learning based technique.

In [WZA06], Williams et al test and evaluate a number of machine learning algorithms and feature selection algorithms for traffic classification. They found that there has been a significant difference in building[4] and classification time among different ML algorithms. The majority of the selected ML algorithms produced very high accuracy (over 90%). It has been reported that the wrapper method provides the best accuracy although it is slow while the filer method is faster but with significantly less accuracy.

In [JMSW07], Jiang et al present a lightweight classification solution based on NetFlow [NF06] data with ML algorithms. In [JEF07], Jeffrey et al propose a semi-supervised classification approach that consists of two components: learner and classifier. The learner is realised by unsupervised means while the classifier is realised by supervised means.

- Unsupervised

A number of classification schemes using unsupervised learning techniques like Machine Learning and clustering have appeared recently.

In [BTA06], Bernaille et al develop an unsupervised clustering (K-means) based classifier with the early classification ability. The early classification means that a classifier is able to identify a network application during early stage of its life cycle. The classifier presented in [BTA06] is dedicated to classify TCP traffic and only takes one parameter into account for classification, which is packet size distribution. Although the overlap issue (i.e. between POP3 and SMTP) among clusters has been reported, no solutions have been presented.

[4] It means the time spent to construct an ML algorithm.

In [EAM06], Erman et al evaluate three unsupervised clustering techniques K-Means, DBSCAN and AutoClass for classification and give related comparison results in terms of classification accuracy and efficiency. In this study, a number of parameters have been used such as the total number of packets of a flow, mean packet size, mean payload size etc. The evaluation results show that the AutoClass algorithm yields the best overall accuracy while DBSCAN requires fewer clusters for traffic classification. One of the drawbacks in [EAM06] is that the classifier is not the early classification based solution in comparison to [BTA06]. For example, it requires the total number of packets of a flow. Secondly, it does not address the feature selection issue and the overlap problem.

Both [ZNA05] and [MHL04] use the AutoClass algorithm for classification. The wrapper based feature selection has been applied in [ZNA05] and [MHL04]. But the impact of using the wrapper based feature selection on the classifier results has not been reported.

In [ZAG06], Zhou et al develop a clustering based classification scheme that is able to produce attack traffic patterns according to the frame management distribution features in the 802.11 WAN environment.

2.5 Discussions and Comparisons

2.5.1 Discussions

Both port-based and payload-based classification can be regarded as rule-based solutions, namely through the analysis of some specific rules in known applications as shown in Table 2.1. The drawbacks and weakness of rule-based classifications are apparent: while

on the one hand requiring considerable manual analysis and on the other hand failure to detect new emerging unknown traffic classes.

Table 2.1: Different rule-based network classification schemes

Inputs	Prior knowledge	Rule-based algorithms
port numbers	checking registered ports	rule-based comparison with port number
payload: signature	to know signature value	comparison with pre-known signature
payload: protocol	to learn protocol value	comparison with pre-known protocol
traffic content (header + payload)	analysing specific traffic (i.e.P2P)	Comparing the results gained from hybrid algorithms

Classification through the analysis at transport-layer is an effective approach to detect P2P traffic. But this approach is problematic when required to detect a large number of network applications unless it combines other classification approaches and uses other information like packet size, flow duration etc [KPF05].

The statistics based classification approach is gaining popularity as it needs less human involvement, for it takes advantages of the advanced techniques like Machine Learning and K-means clustering. Such a scheme is not only able to detect known network applications but also to distinguish unknown traffic classes.

2.5.2 Comparisons of the proposed classifier with relevant literature

The proposed classifier in this thesis belongs to the statistics based traffic classification scheme, which is independent of payload protocols and any signature information. It is close to the studies in [BTA06][DBL03][MZ05][ZM05][EAM06][ZNA05] [MHL04].

It is made with the same assumption as [MZ05] that each discriminator follows a Gaussian distribution. However unlike the previous work based on ML algorithms, this thesis is going to investigate the potential of using Kohonen Neural Network (KNN) to fulfil the

unsupervised learning task. The underlying reason for choosing KNN is because its core algorithm is Gaussian-based. As a sequence, KNN has got the natural mapping and learning capabilities required for classification. In comparison to other unsupervised means like clustering techniques, KNN provides the powerful two-layer learning structure that is able to transform varying traffic profiles into different patterns with the natural mapping and better forms of shapes. KNN will be fully discussed in Chapter 3.

This thesis goes on to investigate the use of the Principal Component Analysis (PCA) to seek the best optimal feature selection for traffic classification. In addition to that, it takes a further step to tackle the overlap issue with the use of Linear Discriminant Analysis (LDA) since this problem has not been discussed in the current literature.

The proposed classifier has the merit of being an early classification solution. Up to now, most of the current classifiers are unable to perform early classification except [BTA06]. Early classification is vital to many management tasks since late classification imposes a number of issues. First, late classification requires considerable processing load and storage memory. Secondly, it is too slow for real-time tasks i.e. rerouting, flow prioritising etc.

2.6 Chapter summary

This chapter discussed the motivations for traffic classification: from allowing an active QoS approach to be realised to help tackle fraudulent traffic usage issues. In addition, network optimisation functionalities need to be considered and for example the SATO approach in Ambient Networks requires classification support. This chapter has presented a critical review of relevant prior and ongoing solutions and techniques of traffic classification; these are divided into four classes according to the input parameters being

used. The features of each technique including relevant problems and weaknesses have been addressed. Last but not least, a brief comparison between the proposed classifier and other relevant literature has been made.

Through the review in this chapter, it can be seen that none of the current classifiers is able to reach 100% confidence in terms of classification accuracy. Given the fast rising number and variety of the new emerging applications, a statistics and unsupervised based classifier needs more investigation when it comes to fast determination of unknown traffic categories. The next two chapters go further to explore a classifier with the use of KNN for unsupervised learning, PCA for feature selection and LDA for solving the overlap problem.

Chapter 3

IP Flow Profile Based Automatic Traffic Classification

3.1 Introduction

The last chapter has discussed that the current major classification schemes are categorised into four classes according to what kind of input parameters are being measured. It is noted that the relevant classification outcomes vary considerably. For instance, a classifier based on port numbers proves to be unreliable with only 50-70% accuracy while most of the statistics based schemes can reach over 85% accuracy. The latest studies like [MP05] have shown that the classification accuracy can be dramatically improved if the input information has been sufficiently taken into account. Since the spectrum of input parameters plays such an important role in classification, it is essential to examine how to best describe and organise the rich input data for traffic classification.

Firstly, this chapter borrows the context concept [DEY01] to describe input information for traffic classification. As will be discussed later, the context concept holds a broader meaning in comparison to the current terms like ports and network diameters and effectively allows all the related information to be used for traffic classification. Secondly, this chapter proposes the flow profile concept to organise all the potential contexts at the flow-level. To fulfil the automatic traffic classification purpose, this chapter proposes the use of five contexts to construct a flow profile; to obtain them only requires IP header information and hence they are fast and easy to obtain. The chapter goes on to address why the Kohonen Neural Network is an effective and suitable technique for the purpose of automatic traffic classification. In addition, this chapter evaluates the proposed automatic

classification scheme through a number of proof-of-concept experiments and raises the need for feature selection, which will be discussed in Chapter 4.

3.2 IP Flow Profile

3.2.1 Basic ideas

As discussed in the last chapter, Internet users often deliberately employ different techniques like masquerading to let private network applications pass through firewalls. However, these techniques will not change the original service features of applications i.e. the characteristics of the audio or video. As a result, relevant traffic profiles that reflect service features will also not be changed. Hence, an alternative approach to identify network applications is to identify the distinct features of traffic profiles that network applications generate.

The traffic profile of an application can be described at the packet level or the flow level. It has been stated in [FRE01][Zhao04a][Zhao04b] that the flow-level traffic features often are more stable than those at the packet-level . Hence, in this chapter, the use of the IP flow profile based information for classification is proposed. For the purpose of this thesis, IP flow profile is defined here as follows.

Definition 1: IP Flow profile is defined as the high-level description and representation of traffic features at the flow level.

The next sections will discuss how to use the context concept to describe IP flow profile.

3.2.2 IP flow profile described with the context concept

A variety of terms used to describe inputs for traffic classification have appeared recently such as port number, discriminator[MZ05], network diameters [MC06] etc. These terms are either too narrow or too specific. This thesis uses the concept of context to describe IP flow profile as *context* has a broader meaning in comparison to others. The concept of context has been widely used in many fields [FR00][DEY01], and is defined as any information or knowledge that can be used to customize the situation of an entity [SCH99][SCH01][BHL01]. For instance, an entity can be one specific IP flow and the associated contexts can be such as a flow's five-tuple, packet size and the number of network links etc. Context can be categorised as either explicit or implicit context [KLL06]. Explicit flow contexts refer to information which can be directly obtained such as transport port number. By contrast, implicit flow contexts do not exist inside a flow but rather need additional computation and are derived using other methods, such as the statistical method used in [ZM05].Note that both explicit and implicit flow contexts are useful for classification. The implementation of contexts is realised by network context sensors, which are discussed later.

Flow-centric context paradigm

As discussed in section 2.4, different layers of information have been used for classification. But, this type of information has often been used in an isolated way from one approach to another such as application layer data (for payload-based approach), transport layer (for transport-layer approach), network layer (for port-based approach) etc. Using the context term, such boundaries can be solved. Fig 3.1 presents a flow-centric context paradigm showing a wider view in terms of the potential input data that could be used for classification. The paradigm includes at least three classes of contextual information, which all exert some degree of influence on traffic profile. The first class of contexts are user-related contexts such as user preference, device profile and location information etc. For example, an application traffic rate may be reduced when a user

switches terminal from a personal computer to a mobile handset. The second class of contexts are network-related contexts like jitter and delay information. For instance, TCP based traffic is quite sensitive to network conditions like delay and available bandwidth, since TCP implements the congestion control functionalities. The third class involves traffic based contextual data such as packet size, time interval between packets and the ratio between the number of incoming packets and the number of outgoing packets etc.

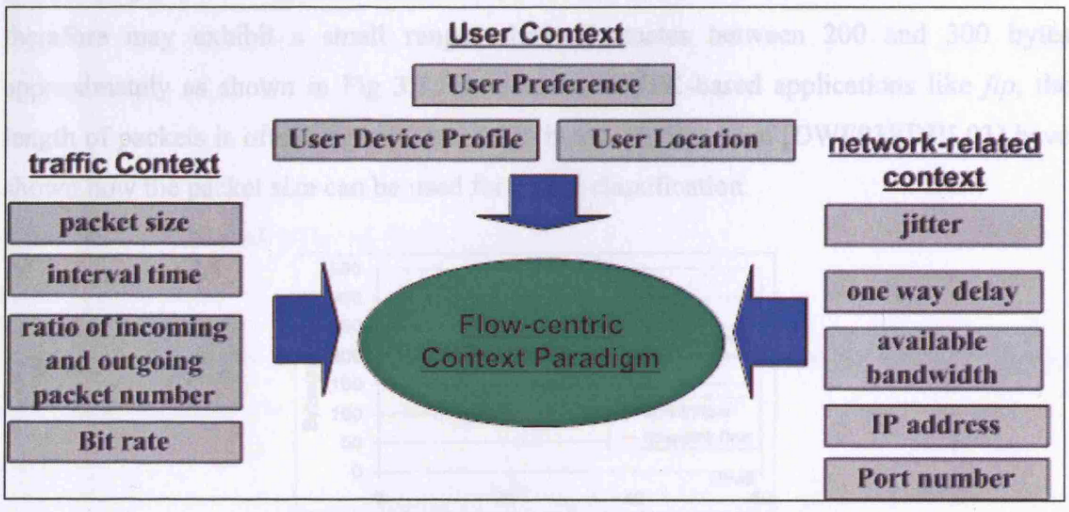


Figure 3.1: Flow-Centric Context Paradigm for Traffic Classification

3.2.3 IP flow profile modelling with IP header based five contexts

Clearly, it may be difficult to collect all the contexts as listed in Fig.3.1 as the collection may largely depend on the physical deployment of a traffic classification system. For instance, it is hard for network operators to obtain the user-related contexts at the core network nodes. In this chapter, a number of IP header based contexts are proposed to serve as the objectives of automatic traffic classification, since they are easy and fast to collect, as discussed below.

Packet size

The packet size refers to the number of bytes in a packet and is usually counted at layer 3. According to TCP/IP protocols, the range of fragmented layer 3 packet size is between 40-1500 bytes in Ethernet. In general, the length of a packet is assigned by the application level protocols and therefore it reflects service features to some degree. For example, the packet length of a Skype flow is mainly determined by the audio codec settings, and therefore may exhibit a small range which fluctuates between 200 and 300 bytes approximately as shown in Fig 3.2. By contrast, BULK-based applications like *ftp*, the length of packets is often very close to 1500 bytes. Studies from [DWF03][DBL03] have shown how the packet size can be used for traffic classification.

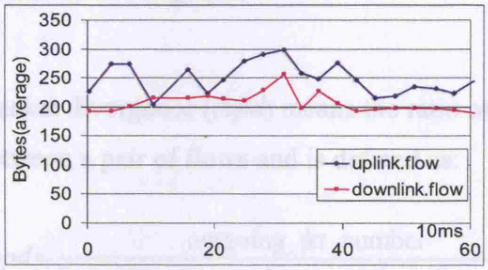


Figure 3.2: Packet sizes from Skype's voice flows

Time gap

The time interval between two back-to-back packets is called the time gap. It is largely determined by the application level requirements. Fig 3.3 shows that the time gaps[5] for Realplayer are more constant than those in Skype's traffic which is paced by the human voice.

[5] The unit for the time gap as shown in y-axis is x10 microseconds.

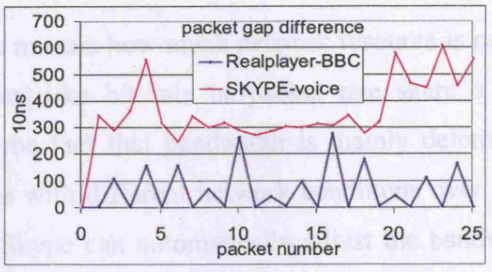


Figure 3.3: Time gaps of RealPlayer and Skype

Other network conditions such as jitter[6] and application level protocol (e.g. audio codec) also affect the values of time gaps.

Flow bidirectional packet divergence

The flow bidirectional packet divergence (*fbpd*) means the ratio of the number of outgoing and incoming packets between a pair of flows and is defined as:

$$fbpd = \frac{\text{outgoing\_pkt\_number}}{\text{incoming\_pkt\_number} + \text{outgoing\_pkt\_number}} \tag{3.1}$$

Fbpd can provide information about the traffic nature of a traffic class. For instance, GAME applications often produce symmetric traffic in both directions (incoming and outgoing) and therefore related *fpbd* values often approach 0.5. However, BULK applications often yield asymmetric traffic. The transport layer protocols also exert influence on *fpbd* values. Classical examples are TCP and UDP: TCP based applications produce more traffic in both directions than UDP-based applications, since TCP requires return packets for acknowledgement and error reports.

Bandwidth of flows

[6] Jitter means the variation of the delay of the received packets.

Bandwidth (*bw*) of flows mirrors how much network resource is needed by the associated application. Other metrics like bit rate or packet rate share a similar meaning with bandwidth. Apart from the fact that bandwidth is mainly determined by the nature of application, it also varies with different network conditions over time. For example, the in-built audio codec of Skype can automatically adjust the bandwidth so as to adapt to changing network conditions.

Evolution of flows

The shape of a flow is not static but dynamically changing. A flow's evolution describes such changes and shape is defined as the sinusoidal value of the angle between packet length and time gap (Fig 3.4):

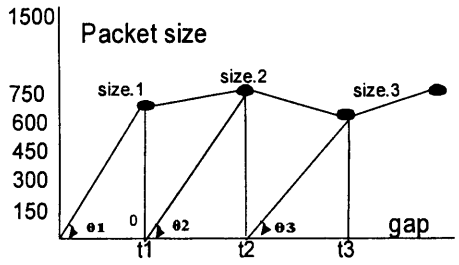


Figure 3.4: The simplified flow evolution diagram

(Note that θ in this diagram is used to give a visual view how the shape is affected by packet sizes and time gaps. But, to calculate the sinusoidal value, the gap needs to be transformed into the same unit as packet size. Hence, the θ in this figure is actually different from θ in the Equation 3.2)

$$flow\_trend = \sin(\theta) = \frac{length}{\sqrt{length^2 + (gap \times bw)^2}} \tag{3.2}$$

where bw, the bandwidth of the associated physical link, is used to transform the time gap into the same unit as packet size. A flow's evolution value increases if $f'(length) > f'(gap \times bw)$ and decreases otherwise.

TCP Congestion Window

It should be mentioned that TCP congestion window size exerts a great deal of influence on the evolution of TCP based flows. This can be illustrated as follows. According to TCP, in Reno's algorithm [FF96][RFC2001], if the *cwnd* reaches the threshold value (exponentially) at the Slow Start stage, the *cwnd* will be set to 1 while the threshold value will be halved. Afterwards, it will take φ steps to reach the previous threshold again (Equation 3.3). Let n refer to the step number that TCP will take to reach threshold value during the first iteration and φ stand for the step number that TCP will reach threshold after the first iteration. For example, if the slow start threshold (*sst*) value is *sst=16*, TCP takes about *n=4* steps to reach the threshold as shown in Fig 3.5. After that, in the ideal network conditions, it takes 3 steps in the slow start stage and 8 steps in congestion avoidance stage (AIMD)[7] before reaching threshold again.

$$\varphi = \log_2^{2^{n-1}} + 2^{n-1} \tag{3.3}$$

and the associated AIMD ratio equals:

$$AIMD\_ratio = 2^{n-1} / (\log_2^{2^{n-1}} + 2^{n-1}) \tag{3.4}$$

Since *sst* varies in different TCP flows, the time spent on AIMD is different as shown Table 2. This implies that the *cwnd* size largely affects the changing pace of TCP flows.

[7] It is also called AIMD- Additive Increase/Multiplicative Decrease.

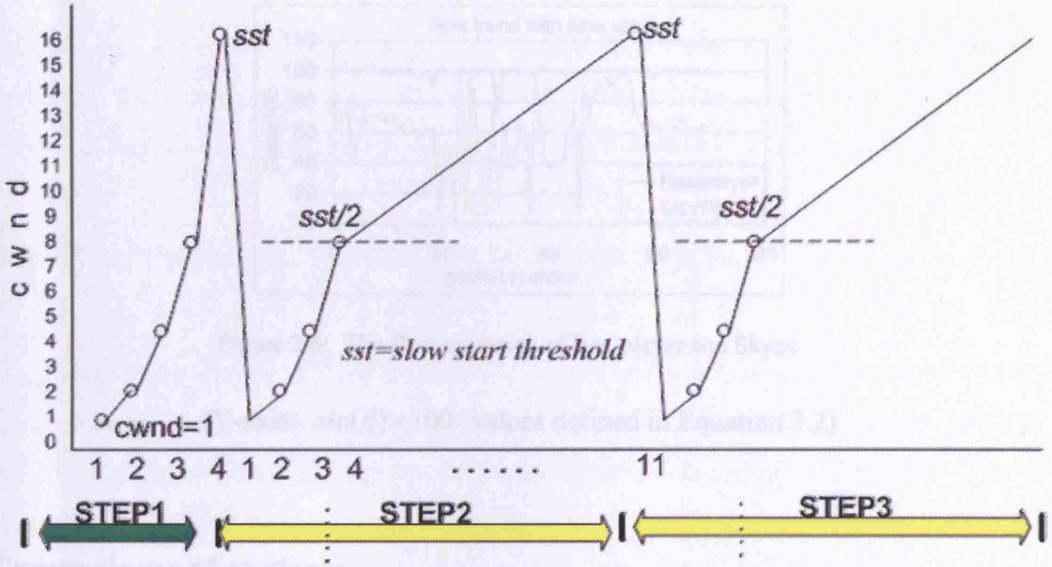


Figure 3.5: A simplified *cwnd* evolution diagram at Slow Start

(Y-axis: cwnd size; X-axis: the step number that TCP will take to reach *sst*)

Table 3.1: TCP's *AIMD* ratio with different threshold values

sst value	n	φ	AIMD ratio
16	4	$3+2^3=11$	8/11=0.7272
32	5	$4+2^4=20$	16/20=0.80
64	6	$5+2^5=37$	25/3=0.865

Fig 3.6 shows that a flow's evolution is closely connected with its network application. It is clear that the flow evolution trend value of Realplayer is notably distinct from Skype's. The former stays more stable while the latter appears with quite large fluctuations.

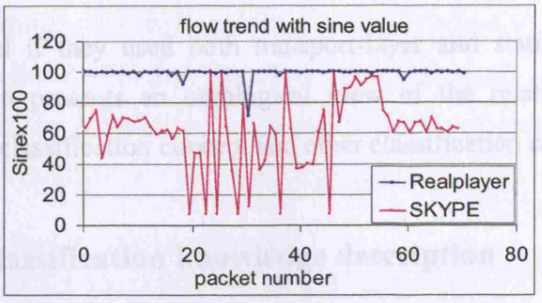


Figure 3.6: The flow evolution of Realplayer and Skype

(Y-axis: $\sin(\theta)\times 100$ values defined in Equation 3.2)

Comparisons of contexts

To start with, the calculations of all the five contexts are mainly based on IP header information, and, all the five contexts relate to the nature of the applications. Nevertheless, there exist varying degrees of correlations between them in which some are strong while others are weak. For example, although bandwidth and evolution of flows seem to offer a better geometric view in term of the shape of IP flow profile, they may correlate with the first three contexts. Therefore, the selection of which contexts should be used for classification is essential. More discussions regarding the input selection will be discussed in Chapter 4.

3.3 The ontological picture of the IP flow profile classification concept and other concepts

As discussed in section 2.4, there are a number of different schemes available for traffic classification. They are realised with varying detection concepts such as payload decoding, signature scanning, traffic statistic analyzing etc. It is often that a classification system may require different classification concepts integrating together to achieve high accuracy. For example, in [KPF05], Karagiannis et al demonstrate that the detection accuracy of

BLINC was improved if they used both transport-layer and statistics-based detection concepts. This section presents an ontological view of the relationship between the proposed flow profile classification concept and other classification concepts.

3.3.1 The traffic classification knowledge description

Recalling Table 2.1 from Chapter 2 which lists a summary of a variety of detection solutions and each solution contains at least two types of information. One type includes the kind of input data and any requirements for prior knowledge; the other type of information is the detection algorithm itself. For example, a payload-based technique typically needs information on how many bits are used for signature, what the signature value is for a particular application. The traffic detection algorithm is then realised by comparing input data with the prior knowledge. Therefore, we need at least two mechanisms to fully represent the knowledge enshrined in a traffic detection system. The input data, as discussed above, can be achieved through the use of the context concept. The other is used to describe algorithms on traffic detection, which can be achieved through the use of ontologies.

Ontology has been extensively used for various knowledge management purposes in different areas such as Semantic Web [BHL01], flow classification [OGT06], semantic routing in P2P networks [CAS03] [SSD02] etc. Gruber defines a specific ontology as the explicit specification of conceptualisations used to help programs and humans share knowledge [GRU93a] [GRU03b]. In general an ontology includes structured knowledge statements that describe the concepts of a domain and their relationship [ZZP06] [SHJ06] [YZX06]. Another important term related to ontology is the knowledge base, which is formed by concept instances. There is a direct connection between context and ontology. On the one hand, as stated in [SGB00], the main components of ontology are the domain concepts. Both concepts and concept instances are called entities [SHJ06]. On the other hand, according to the context definition, contexts are defined as any information used to characterise such entities. Hence, the connection between context and ontology is linked

by entities. Correspondingly, a simple ontology based structure to describe classification knowledge can be shown in Fig 3.7. The bottom of this figure is all about inputs for classification while the top left part is for the algorithm description of different concepts and the top right part is practical instances of ontologies.

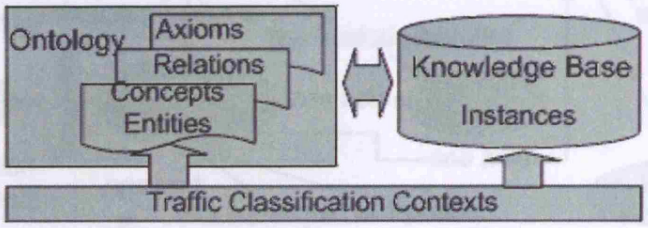


Figure 3.7: The relationship between context, ontology and knowledge base

3.3.2 The ontological picture of the traffic classification concepts

It is interesting to see the position of the proposed flow profile diction concepts in the whole classification ontological picture. Naing et al. [NAI02] define an ontology that consists of six elements: $\{C, A^C, R, A^R, H, X\}$, where C refers to a set of concepts; A^C represents attributes for each concept; R represents a set of relationships; A^R represents the attributes for a set of relationships; H stands for a concept hierarchy and X represents a set of axioms. According to this definition, the traffic classification ontology is presented in Fig 3.8, which mainly focuses on concepts C and *subClassOf* relationships R. In this ontology, *network* is root and includes two concepts: *nodes* and *traffic*, where, in *traffic*, three concepts are exploited: *traffic category*, *IP flow profile* and *traffic statistic.*

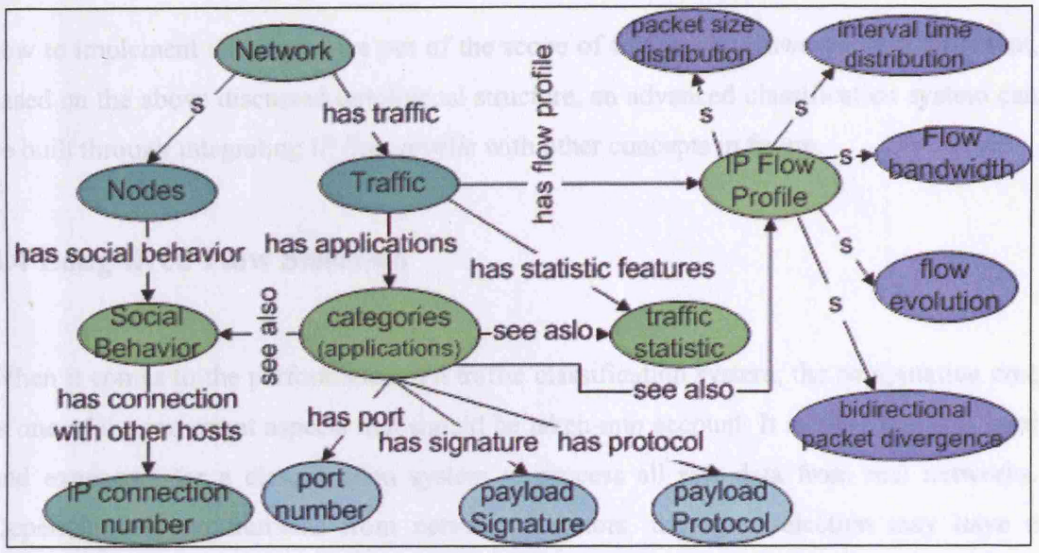


Figure 3.8: Traffic classification ontology (s= *subClassOf*)

As shown in Fig 3.8, the '*application'* has various relationships with other concepts: *payload signature, payload protocol, traffic statistic, IP flow profile* and *social behaviour*. The existing major detection concepts discussed in section 2 have been included in this ontology such as *layer 4 port, payload protocol, traffic statistic, IP flow profile, social behaviour* etc. It should be noted that here the contexts share the same meaning with A^C. For example, for the concept '*layer 4 port*' the contexts can be TCP or UDP ports, which are further divided into three classes[8]: well-known port numbers [0-1023], registered port numbers [1024-49151] and unregistered port numbers. Similarly, there exist different sets of contexts for each detection concept.

As it can be seen, the proposed *IP flow profile* concept is a child of the *traffic* concept and connects to *traffic categories,* through which it links to all other classification concepts. A number of languages are available for the ontology implementations like OWL (Web Ontology Language) and OWL-DL (Description Logics) [BHH04]. In [OGT06], the ontology implementation of a context-aware flow classification has been given. Details of

[8] http://www.iana.org/assignments/port-numbers

how to implement ontologies are out of the scope of this thesis. However, it is clear that, based on the above discussed ontological structure, an advanced classification system can be built through integrating *IP flow profile* with other concepts in future.

3.4 Long-lived Flow Selection

When it comes to the performance of a traffic classification system, the computation cost is one of the important aspects that should be taken into account. It is clear that it is hard and expensive for a classification system to process all raw data from real networks. Depending on requirements from network operators, the flow selection may have a significant impact on reducing classification computation cost. For instance, if a classification system is used to provide services such as QoS, adaptation of networks and traffic engineering, it is often sufficient to classify long-lived flows rather than all flows.

3.4.1 Long-lived flow vs. short-lived flow

According to the flow duration and the associated extent of traffic, a flow can be categorised into either the long-lived flow (elephant) or short-lived flow (mice). It has been shown that 20% of the flows have more than 10 packets but these flows carry 85% of the total traffic [SRS99] [DOW01]. As shown in Fig.3.9, the solid lines representing the flows that have over 10 packets drop dramatically while the dashed lines reflecting relevant packet proportions decrease slowly.

Figure 3.9: The long-lived flow and short-lived flow distributions [SRS99].

(In this Figure, a flow is categorised as long-lived flow if its length is over 10 packets)

As the proposed classifier is an early detection scheme, the cost to classify a long-lived flow and a short-lived flow is identical. Therefore, if a classification system is only required to classify the traffic that belong to LLFs, the classification cost can be significantly reduced (almost 80% reduction of the total cost) since 20% of LLFs accounts for over 80% network traffic while 80% of short-lived flows accounts for only about 20% network traffic. It should be noted that the detection of a long-lived flow itself adds extra complexities to a traffic classification system, which is discussed next.

3.4.2 Long-lived flow selection

A few mechanisms have been proposed to isolate long-lived flows from short-lived flows. In [YM01], Yilmaz and Matta proposed the solution to classify LLF and SLF by evaluating the burstiness (β) of TCP packets. They stated that TCP sends a burst of packets every round-trip time (RTT) since it is window-based. They assumed that TCP's arrival followed the Poisson process with the rate λ_{TCP} and corresponding burstiness

equalled $\lambda_{TCP} \times RTT$ within every RTT. If β>4 belongs to LLF and otherwise SLF[9]. This is because TCP congestion windows of LLFs are generally larger than those of SLFs [MG00]. This phenomenon happens because LLFs spend most of time in the *Congestion Avoidance* phase but SLFs mainly stay in the *Slow Start* phase.

[NLM96] proposes the packet-count flow classifier based on *X/T* algorithms. *X* refers to the packet arrival number of a flow while *T* means the timeout value. When a flow's *X* exceeds a given threshold within *T*, this flow is categorised into a long-lived flow. As parameters *X* and *T* are assigned with static values in [NLM96], Hao et al argue that its result will not be accurate when the *X* value is decreased in a congested network connection [CLL98]. The follow-up studies from [WCS01][LC01] propose the X/Y/T algorithm which allows X to be adaptively adjusted according network state and the utilisation of resource within the selected time Y.

As it can be seen from above, the long-lived flow detection like X/T or X/Y/T is quite straightforward to apply for traffic classification. There may exist other algorithms that offer better granularity than the above algorithms. But, for traffic classification, even if the long-lived flow detection granularity is slightly coarse (e.g. 10% of short-lived flows have been misclassified as long-lived flows), the classification cost still can be dramatically reduced.

3.5 Towards Automatic Traffic Classification

As will be discussed below, the key to achieving automated traffic classification abilities is how to transform a high-dimensional IP flow profile into the two-dimensional map, a map that is automatically produced through the unsupervised learning algorithms like Kohonen Neural Network. This section firstly discusses the relationship of an IP flow profile to its traffic class. Next, it discusses the distribution feature of an input context,

[9] Note that β=4 is only an empirical value.

which ideally should be Gaussian-based. Then, it analyses why and how KNNs can be used to construct IP flow profile maps.

3.5.1 Inherent relationship of an IP flow profile to its traffic class

As discussed in section 3.2, each flow profile is described by a set of contexts $\{C\}$ and each context $C_i : \{C_i, i = 1...N\} \subseteq C$, where N is the number of contexts. In some cases, an application can be easily recognised by a single context value. One of the simplest examples is *ftp*, which is associated with the port number (21). Another example is given in [DBL03], which shows that a series of network applications may be determined by a single context - packet size distribution. Nevertheless, it is more often the case that the network application detection requires multiple context inputs. For example, by analysing *fpbd* values, GAMES applications can be distinguished from MULTIMEDIA applications[10]. If the traffic is determined as MULTIMEDIA application, a further set of contexts *{packet size, time interval}* can be used to identify whether it is primarily an audio or video based application. Since each application has its own IP flow profile features, as discussed in last section, a corresponding set of contexts always exists. In addition, classification accuracy is increased, after each regrouping step among C_i, and, as a consequence, distinguishability between different IP flow profiles should improve but may not always be unique. Clearly, to understand which contexts should be used and how to regroup these contexts is complex and is difficult to be realised through the manual analysis approach. The next sections describe how the use of Kohonen Neural Network allows a regrouping of the contexts in a systematic and reliable manner.

[10] We assume there are only two applications in the target traffic: games and multimedia.

3.5.2 The mathematical characteristics of each context input

In the one-input case, if the context input possesses a fixed value during a flow life cycle, the classification is often implemented by the rule based checking method like the port-based solution discussed in section 2.4.1. While its value is dynamic, the traffic classification can be realised through analysis of its distribution features. The studies of [MZ05][MP05] have reported that the discriminators for traffic classification follow the Gaussian distribution. This is based on the assumption that the discriminators are independent of each to another [MZ05].

The shape of a Gaussian distribution defined in Equation 3.5 is determined by its mean (μ) and variance (σ^2).

$$f(x \mid \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} \times e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{3.5}$$

Such Gaussian distribution features can be seen from the following diagrams. Fig 3.10 (a) shows the distribution of Realplayer's packet sizes. In this figure, the width of each bar is equivalent to 20 bytes while the height of each bar represents relevant frequency. The red line is produced with the Kernel Density Function [11] (see Appendix C.1 for implementation details) and this line shows that it is close to a Gaussian distribution. In addition, as can be seen, the packet sizes of Realplayer are mainly in the range from byte 400 to byte 800. Note that in practice, some context distributions may not be exactly Gaussian. For example, as shown in Fig 3.10-(b), the distribution of Skype's packet sizes does not follow a single distribution as it is mainly centred at bytes 10, 50 and 100. This will affect the classification results if using the Gaussian-based classification algorithm; this will be discussed in section 4.6.1.3.

[11] Using the default bandwidth (equal to 22.4187 bytes) of ksdensity( ) function with Matlab.

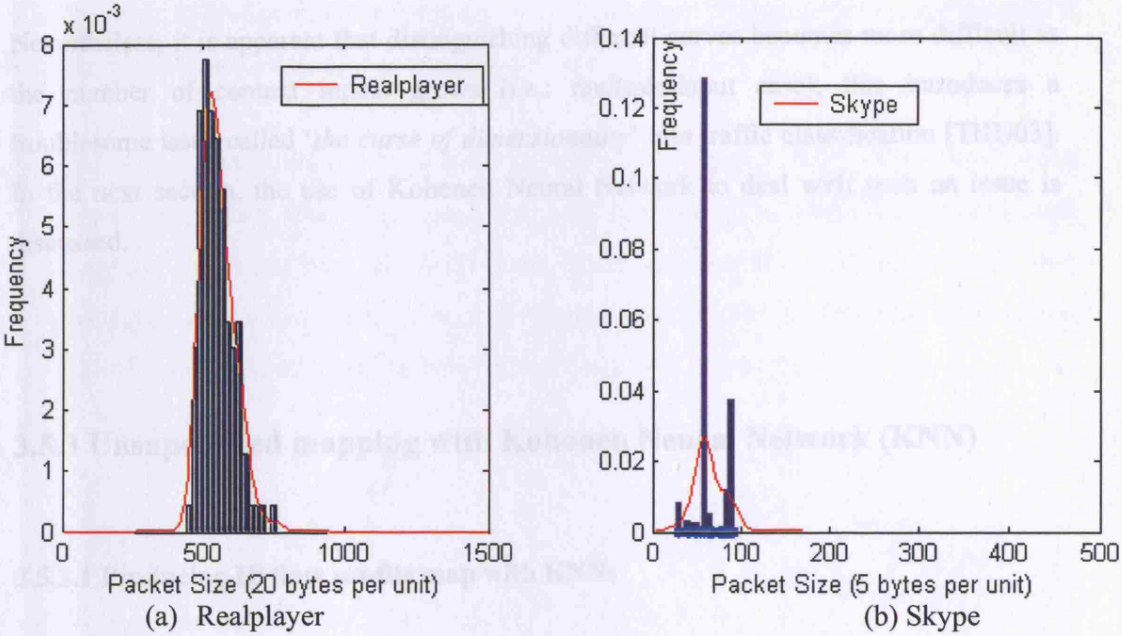


Figure 3.10: The packet size distributions

An IP flow profile can be labelled according to its unique distribution feature. For example, in Fig 3.10 (c), the two Gaussian curves (20,5), (55,12) where these represent the couples (μ , σ) represent the context distributions generated from two different applications. Obviously, the less overlap between the curves leads to better detection results (the better distinguishability between such contexts).

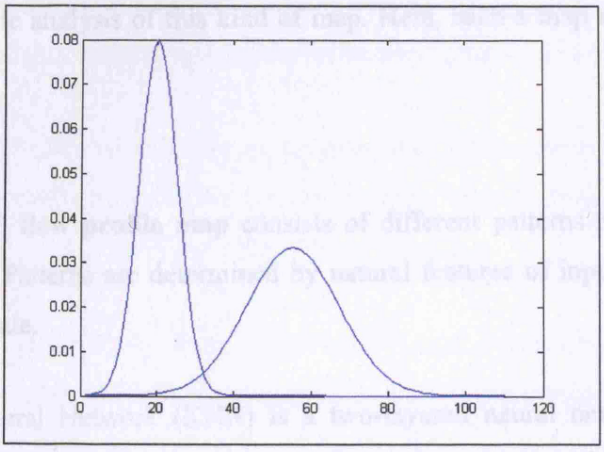


Fig.3.10 (c): Two different Gaussian curves

Nevertheless, it is apparent that distinguishing different curves becomes more difficult as the number of context inputs grows (i.e.: multiple-input case); this introduces a troublesome issue called '*the curse of dimensionality*' into traffic classification [THU03]. In the next section, the use of Kohonen Neural Network to deal with such an issue is discussed.

3.5.3 Unsupervised mapping with Kohonen Neural Network (KNN)

3.5.3.1 Producing IP flow profile map with KNNs

The preceding section discussed two important factors that involve traffic classification.

- *Each context input is independent from one to the other and as a consequence, it follows a Gaussian distribution.*
- *There exist 'the curse of dimensionality' issue in the multiple-input case.*

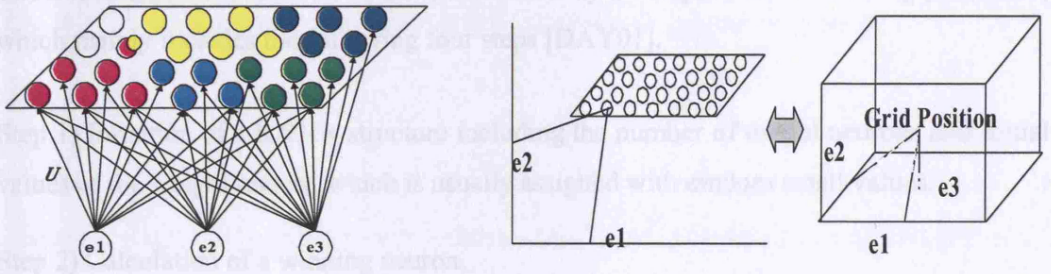
To deal with the issue of *'the curse of dimensionality'*, we need to transform an IP flow profile into a low dimensional map i.e. 2-D. Then the traffic classification task can be realised through the analysis of this kind of map. Here, such a map is called an IP flow profile map.

Definition: **An IP flow profile map** consists of different patterns representing distinct traffic categories. Patterns are determined by natural features of input data that network applications generate.

The Kohonen Neural Network (KNN) is a two-layered neural network (Fig 3.5.2-a), which is also called a Kohonen's self-organising map or Kohonen's feature map [KOH01].

It is proposed to be a good basis for producing the required IP flow profile maps because it handles both of the two factors as discussed below:

1) First of all, one of the principal goals of KNN is to map a high (n) dimensional input space $\Re^n$ onto a low-dimensional (i.e. 2-D) grid map [DAY90] (Fig 3.11-b). Such mapping functionalities of KNN have been successfully applied to deal with categorisation issues such as speech recognition [KOH01], semantic information classification [LSM91] and Internet categorisation [CSO96] etc.



(a) (b)

Figure 3.11: Kohonen Neural Network architecture with two or three inputs

2) Secondly, KNN's learning ability is developed according to the relationship between input signal vectors $E \in \Re^n$ and weight vectors $U \in \Re^n$ with the Gaussian-based competitive learning algorithm [KOH01]. This particularly matches the mathematical feature of each input context which also follows the Gaussian distribution.

3) In addition, KNN is realised with an unsupervised self-organising manner and does not needs the target outputs as the penalty to adjust the weight vectors. Such features add automatic[12] capability into a KNN-based classifier.

4) What is more, the maps produced by KNNs give ideal visualisation results in the form of different patterns with nature shapes (Fig 3.11-a). As stated in [LSM91][LO92],

[12] Since it is unsupervised-based approach

KNN forms clusters (patterns) that match better to the desired classes than the classical K-means that only produces spherical shapes.

5) Finally, it should be mentioned that, the author of the Ph.D thesis [MIK05] also has suggested that KNN should be investigated for traffic classification in future.

3.5.3.2 Gaussian-Based Kohonen Neural Network Algorithms

The above explanations can be further illustrated by a complete KNN learning procedure, which mainly includes the following four steps [DAY01].

Step 1) Initiation of a KNN's structure including the number of output neurons and initial values of the weight matrix, which is usually assigned with random small values.

Step 2) Calculation of a winning neuron.

Let $E(t) = \{E_1(t), E_2(t), ..., E_N(t)\}$ be the N-dimensional input vector at the t[th] learning time and $U^{(i)}(t) = \{U^{(i)}{}_1(t), U^{(i)}{}_2(t), ..., U^{(i)}{}_N(t)\}$ the weight vector for neuron i at the t^{th} learning step. Then, KNN starts to compute a matching value for each neuron in the grid map. For example, the matching value of neuron i is equal to:

$$\left\| E(t) - U^{(t)}{}_i \right\| \tag{3.6}$$

Then calculate the Euclidian distance between inputs and neuron i:

$$distance = \sqrt{\sum_j (E^{(t)}{}_j - U^{(t)}{}_{ij})^2} \tag{3.7}$$

Now repeat the above calculation for every neuron, a wining neuron is the one (k) that has the lowest Euclidian distance value:

$$\left\| E(t) - U^{(k)}{}_t \right\| = \min_i \{ \| E(t) - U^{(i)}{}_t \| \} \tag{3.8}$$

Step 3) Updating the weight matrix

In this step, KNN adjusts the neighbourhood's weights according to the following equation.

$$U_{ij}^{(t+1)} = U_{ij}^{(t)} + \frac{dU^{(t)}_{ij}}{dt} \quad \text{subject to:} \quad \frac{dU^{(t)}_{ij}}{dt} = \begin{cases} \alpha(t)(E_j(t) - U^{(t)}_{ij}) & j \in N \\ 0 & otherwise \end{cases} \tag{3.9}$$

where $\alpha(t)$ is the updating rate and $U_{ij}^{(t+1)} = U^{(i)}{}_{j(t+1)}$, which stands for the weight value from the j^{th} input to the i^{th} neuron.

Step 4) Repeating step 2 and 3 until KNN converges to the satisfied result.

The updating rate $\alpha(t)$ is updated iteratively during each learning procedure and is realised through adjustments to the learning rate η_t and neighborhood shrinkage rate d_t. A variety of definitions of $\alpha(t)$ exist [KOH01]. For example, in the earlier version of KNN, both η_t and d_t are defined as the linear with the learning times t [DAY90]:

$$\eta_t = \eta_0(1 - t/t_{\max}) \tag{3.10}$$

$$d_t = \lceil (d_0 - t/t_{\max}) \rceil \tag{3.11}$$

However, non-linear based definitions to $\alpha(t)$ have been proven to be more effective and successful in comparison to the linear one, as stated in [LSM91]:

> *"One of the successful stories of current neural network approaches is to apply nonlinear, continuous functions such as the sigmoid function and the Gaussian function to the learning process. The Gaussian function is supposed to describe a more natural mapping so as to help the algorithm converge in a more stable manner"*

It is assumed that the Gaussian-based learning process will be more suitable for traffic classification since input contexts also follow a Gaussian distribution. Therefore, in this thesis, the following formula for the updating rate is adopted.

$$\alpha_t(i,k) = \exp(-t/t_{\max}) \times \exp\left(\frac{-\|r_i - r_k\|^2}{2\sigma^2(t)}\right) \quad (3.12)$$

where node k is the winning neuron. In this formula, the first Gaussian function is to control the learning rate (η_t) and the second Gaussian function deals with the neighborhood shrinkage (d_t) adjustment; $\sigma(t)$ is the width parameter which is reduced as t increases [HAG95] [LSM91]. The definition of $\sigma(t)$ stated in [HAG95] is used in this thesis. More details on Kohonen Neural Network can be found in [KOH88][KOH01] [DAY01] [LSM91][HAG95].

3.6 Implementations

This section illustrates a number of implementations developed in this research for the proposed classification scheme, including how to transform IP flow profile raw data into KNN inputs, the design of IP traffic collection and prototype KNNs.

3.6.1 Inputs for KNN

It is apparent that the values of input data vary widely. For example, the byte size could be any values between 40 and 1500 while the *fpbd's* value ranges from 0 to 1. The high degree of data divergence introduces difficulties for convergence of KNN. Thus, the first step is to reduce the divergence of the acquired input data. One typical solution is to transform the input data into ratio values $r \in [0,1]$. Most inputs can be easily converted by comparing them with related upper bound values. The upper bound of an input can be either assigned by relevant TCP/IP protocols (e.g., the maximum packet size equals 1500) or obtained from experimental data. Table 3.6.1 lists nine ratio values relating to the five key contexts mentioned above and related abbreviation terms are:

in_size and *out_size* refers to the incoming and outgoing packet size (length) respectively;

in_gap and out_gap stands for the incoming and outgoing packet interval respectively;

BW_{IN} and BW_{OUT} denote the physical bandwidth of the uplink and downlink;

MaxT is the threshold value of the interval time while *t* stands for the time length to measure BW of flows.

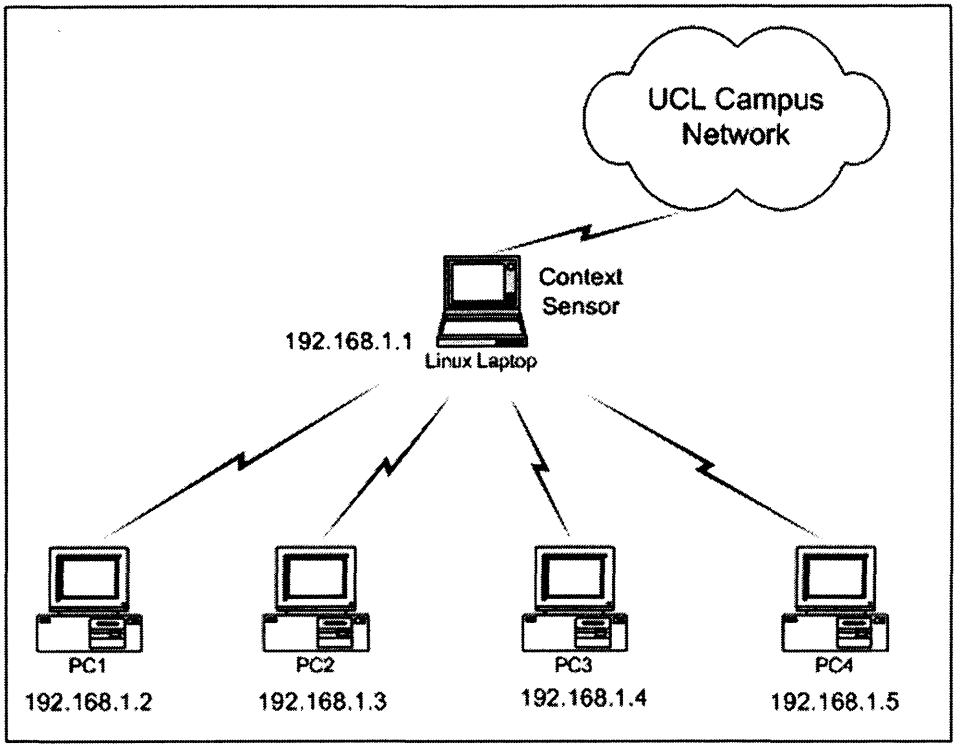
Table 3.2: Input breakdown for KNN

parameter	ratio equation	context type
parameter 1	$Rl = \frac{out\_pkt\_no}{in\_pkt\_no + out\_pkt\_no}$	C3: fpbd
parameter 2	R2=out_size/1500	C1: packet size
parameter 2	R3=out_gap/MaxT	C2: packet gap
parameter 3	R4=in_size/1500	C1: packet size
parameter 4	R5=in_gap/MaxT	C2: packet gap
parameter 6	$R6 = \frac{out\_size}{\sqrt{out\_size^2 + (out\_gap * BW_{OUT})^2}}$	C4: fe (flow evolution)
parameter 7	$R7 = \frac{in\_size}{\sqrt{in\_size^2 + (in\_gap * BW_{IN})^2}}$	
parameter 8	$R8 = \frac{\sum in\_size}{BW_{IN} \times t}$	C5: BW of Flow
parameter 9	$R9 = \frac{\sum out\_size}{BW_{OUT} \times t}$	

3.6.2 Traffic Collection

3.6.2.1 Testbed

To collect IP raw traffic for classification, a LAN based testbed has been set up.



(A LAN with the network address of 192.168.1.0)

Figure 3.12: The traffic classification testbed

A number of machines are attached to this LAN. One of them is for traffic collection with context sensors installed under Linux OS. Its network interface card is set to the promiscuous mode so as to capture all the traffic running on this LAN. Others are used to run different types applications under WinXP OS. There are three modules included in network context sensors: traffic collector, flow aggregator and IP flow profile analyser and all of them are programmed with the C or C++ language under Linux OS.

3.6.2.1 Software for Collection

The software designed for traffic collection is called the network context sensor. It was written in C and C++ and has been tested on Linux OS. It consists of three modules: collection module, flow aggregator and flow profile analysis module.

1) IP raw traffic collection module

First of all, the 'socket ( ) function', which is a part of '*pcap*' libraries from Linux OS, is used to monitor the IP raw traffic, with the following format:

```
int sock_id;
sock_id=socket(PF_PACKET,SOCK_RAW,htons(ETH_P_ALL));
```

The setting is to let 'socket( )' function collect all the raw data at the link layer. The collection part is realised through the use of the 'recv ( )' function.

```
int packet_len;
char buf[10000];
packet_len=recv(sock_id,buf,1000,0);
```

Next is to decode the raw traffic according to the format of TCP/IP protocols. The main target in this step is to filter all the IP data rather than other management message. It is realised through the comparison of the 12th and 13rd bytes with 0x0800 as shown below:

```
unsigned int proto_type;
proto_type=(buf[12]*0xff) | buf[13];
if(proto_type= =0x0800)
ip_decode( );
```

2) Flow aggregator

If the received data belongs to IP traffic, it will be stored with the predefined IPDR (IP detailed record) data structure. Then, the flow aggregator is able to identify varying flows according to related five-tuples.

```
struct   ipdr {
         unsigned char   sip[256];
         unsigned char   dip[256];
         unsigned int    sport;
         unsigned int    dport;
         unsigned int    length;
         unsigned int    times;
         unsigned int    proto_type;
         unsigned int    start_time;
         unsigned char   trafic_direction;
         unsigned int    end_time;
};
```

3) IP flow profile analysis module

The IP flow profile analyser is used to transform original traffic data into the data format which KNN can accept.

3.6.3 Kohonen Neural Network Prototype

To achieve successful learning by the KNN, implementations play an important role. Different versions of KNN implementations have been studied for example [DAY01] [HAG95], the key variance is based on how the learning and neighborhood shrinkage rates are adjusted. In this research, the algorithms from [LSM91][HAG95] are adopted because they are Gaussian-based algorithms.

The prototype of Gaussian-based learning KNNs with Matlab

Fig 3.13 to 3.15 show the three core steps of KNN's implementations developed in this research with Matlab. Step 1 is to compute a weight updating (sigma) matrix that stores the corresponding Gaussian-based adjustment rate matrix for every neuron. For each neuron, the Gaussian-based adjustment rate matrix follows the equation:

$$adjustment\_rate = learning\_rate \times \exp\left(\frac{-|dist|}{neighborhoodsize^2}\right)$$

where 'dist' is the square of Euclidian distance adjusted from line 3 to 6, equal to $(i - coord\_x)^2 + (j - coord\_y)^2$. Both learning rate *alpha* and *neighbourhood size* are gradually reduced after each iteration.

```
% Step 1: to compute weight updating matrix for every neuron
% coord_x, coord_y: a neuron position on the second layer of KNN
% M,N: the size of the second layer of KNN, i.e. 9x9
% alpha: learning rate, which is adjusted accordingly with the training times

for coord_x=1:M              %  line 1
for coord_y=1:N              %  line 2
     for i=1:M               %  line 3
        x_dist=(i-coord_x)^2;      % line 4
        for j=1:N                  % line 5
             xy_dist=x_dist+(j-coord_y)^2;       % line 6
             gaussian_matrix(i, j)=exp(-xy_dist/(NeighborhoodSize^2))*alpha; % line 7
        end                                                                  % line 8
     end                                                                      % line 9
        sigma{coord_x, coord_y}=gaussian_matrix;                % line 10
  end
end
```

Figure 3.13: The 1st step of KNN's implementations

```
% step 2: to calculate the winner
% Tdata - the vector stores input context values
% E: the number of inputs

   for data_no=1:size(Tdata, 1);
        dist_kt=0;
        for index=1:E;
         dist_kt=dist_kt+(Tdata(data_no, index)-U(:, :, index)).^2;  % to add all the vectors
together
        end
        [min_data, winner_no]=min(dist_kt(:));
        coord_y = 1+floor((winner_no-1)/M);                          % to get x, y coordinate
        coord_x = winner_no-(coord_y-1)*M;
```

Figure 3.14: The 2nd step of KNN's implementations

Step 2 is to calculate the winning neuron according to the total sum of difference between inputs and weights that correspond to each neuron. The winner that has the minimum value is labelled and its positions on the second layer are stored in *coord_x* and *coord_y*.

```
% step 3: to update the weight
% U: weight matrix

for index=1:E
    U(:,:,index)= U(:,:,index).*(1-sigma(coord_x,coord_y)) +
                  Tdata(data_no,index)*sigma(coord_x,coord_y);
end
```

Figure 3.15: Figure 3.15- The 3rd step of KNN's implementations

Step 3 is to update the weight matrix after a winner has been located. As the weight updating matrix has been computed during step 1, step 3 only needs to simply load the weight updating data to adjust the weight matrix.

It is interesting to see how the updating matrix is being changed as the training times increases. The following diagram shows a number of updating matrices from an arbitrary 9x9 KNN, which are the results of using the above codes.

1) First of all, it can be seen from the following diagram (Fig 3.16) that each element in *sigma* matrices contains a 9x9 weight-updating matrix.

```
>> sigma

sigma =

    [9x9 double]    [9x9 double]    [9x9 double]    [9x9 double]    [9x9 double]    [9x9 double]    [9x9 double]    [9x9 double]    [9x9 double]
    [9x9 double]    [9x9 double]    [9x9 double]    [9x9 double]    [9x9 double]    [9x9 double]    [9x9 double]    [9x9 double]    [9x9 double]
    [9x9 double]    [9x9 double]    [9x9 double]    [9x9 double]    [9x9 double]    [9x9 double]    [9x9 double]    [9x9 double]    [9x9 double]
    [9x9 double]    [9x9 double]    [9x9 double]    [9x9 double]    [9x9 double]    [9x9 double]    [9x9 double]    [9x9 double]    [9x9 double]
    [9x9 double]    [9x9 double]    [9x9 double]    [9x9 double]    [9x9 double]    [9x9 double]    [9x9 double]    [9x9 double]    [9x9 double]
    [9x9 double]    [9x9 double]    [9x9 double]    [9x9 double]    [9x9 double]    [9x9 double]    [9x9 double]    [9x9 double]    [9x9 double]
    [9x9 double]    [9x9 double]    [9x9 double]    [9x9 double]    [9x9 double]    [9x9 double]    [9x9 double]    [9x9 double]    [9x9 double]
    [9x9 double]    [9x9 double]    [9x9 double]    [9x9 double]    [9x9 double]    [9x9 double]    [9x9 double]    [9x9 double]    [9x9 double]
```

Figure 3.16: The illustration of sigma matrix

2) Fig 3.17 presents the weight-updating matrix in the case the first neuron is the winner whose position is (1,1) on the KNN grid map. It shows that the updating rate is symmetrically distributed around the winner neuron (1,1). The larger the

distance to the winner node, the smaller the assigned updating rate according to Equation 3.12.

```
>> sigma(1,1)

ans =

    0.1000    0.0756    0.0327    0.0081    0.0011    0.0001    0.0000    0.0000    0.0000
    0.0756    0.0572    0.0247    0.0061    0.0009    0.0001    0.0000    0.0000    0.0000
    0.0327    0.0247    0.0107    0.0026    0.0004    0.0000    0.0000    0.0000    0.0000
    0.0081    0.0061    0.0026    0.0007    0.0001    0.0000    0.0000    0.0000    0.0000
    0.0011    0.0009    0.0004    0.0001    0.0000    0.0000    0.0000    0.0000    0.0000
    0.0001    0.0001    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000
    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000
    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000
    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000
```

Figure 3.17: The illustration of *sigma(1,1)* matrix

3) Fig 3.18 and 3.19 give a clearer view of how the updating rate is symmetrically spread around the winner nodes (6,6) and (9,9). Note that if a selected structure is not square based, then the updating rate will not be symmetrically spread in the matrix. This explains why the experiments presented next choose square based structures.

```
>> sigma(6,6)

ans =

    0.0000    0.0000    0.0000    0.0000    0.0001    0.0001    0.0001    0.0000    0.0000
    0.0000    0.0000    0.0001    0.0004    0.0009    0.0011    0.0009    0.0004    0.0001
    0.0000    0.0001    0.0007    0.0026    0.0061    0.0081    0.0061    0.0026    0.0007
    0.0000    0.0004    0.0026    0.0107    0.0247    0.0327    0.0247    0.0107    0.0026
    0.0001    0.0009    0.0061    0.0247    0.0572    0.0756    0.0572    0.0247    0.0061
    0.0001    0.0011    0.0081    0.0327    0.0756    0.1000    0.0756    0.0327    0.0081
    0.0001    0.0009    0.0061    0.0247    0.0572    0.0756    0.0572    0.0247    0.0061
    0.0000    0.0004    0.0026    0.0107    0.0247    0.0327    0.0247    0.0107    0.0026
    0.0000    0.0001    0.0007    0.0026    0.0061    0.0081    0.0061    0.0026    0.0007
```

Figure 3.18: The illustration of *sigma(6,6)* matrix

```
>> sigma (9,9)

ans =

    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000
    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000
    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000
    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0001    0.0001
    0.0000    0.0000    0.0000    0.0000    0.0000    0.0001    0.0004    0.0009    0.0011
    0.0000    0.0000    0.0000    0.0000    0.0001    0.0007    0.0026    0.0061    0.0081
    0.0000    0.0000    0.0000    0.0000    0.0004    0.0026    0.0107    0.0247    0.0327
    0.0000    0.0000    0.0000    0.0001    0.0009    0.0061    0.0247    0.0572    0.0756
    0.0000    0.0000    0.0000    0.0001    0.0011    0.0081    0.0327    0.0756    0.1000
```

Figure 3.19: The illustration of *sigma(9,9)* matrix

3.7 Results and Discussions

This section presents a series of proof-of-concept experiments which aim to demonstrate that the proposed IP flow profile based classification scheme can be effectively realised by KNNs. The first part of this section shows the results of an IP flow profile map if different graphs are used as flow profile maps. The second part of this section presents the testing results to show that although a high-dimensional flow profile can be effectively transformed into a 2-D KNN map, the classification outcomes are highly dependent on the selection of inputs.

Three different network applications have been chosen for the experiments: RealPlayer, MediaPlayer and Pool Game. Both RealPlayer and MediaPlayer belong to the MULTIMEDIA traffic class. The purpose of choosing two such applications is to show that KNN is well able to classify traffic even when the detected applications profiles are analogous to each other.

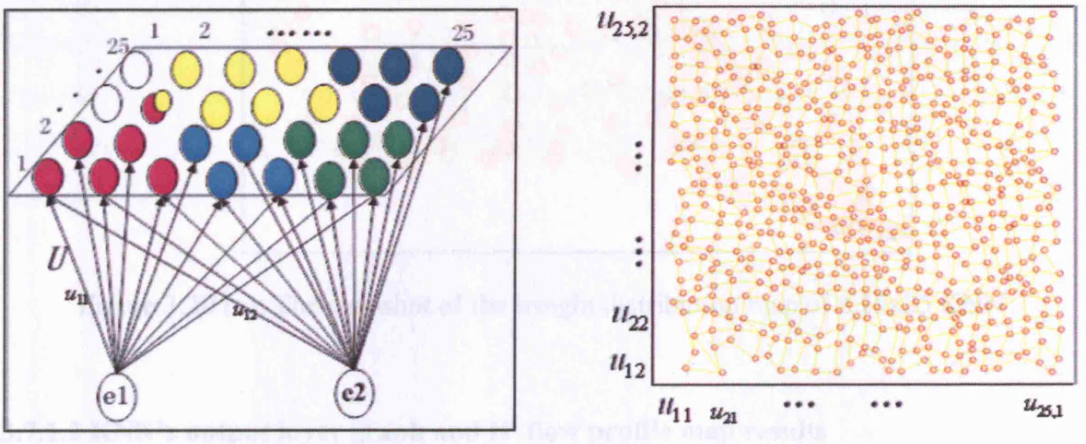
3.7.1 IP flow profile map

Depending on the number of inputs, as will be shown below, there are two methods used to interpret KNN learning results. The following tests compare different results when

using the weight distribution graph and the output layer graphs as flow profile maps. In addition to that, the second subsection also goes on to show how different KNN structures can lead to more reliable classification effects.

3.7.1.1 The weight distribution graph

The KNN learning results can be visualised in a 2-D weight distribution graph when the input number equals two. This method has been discussed in [DAY90][KOH01]. As it can be seen in Fig 3.20 (a), in a two-input KNN architecture, each neuron has two weight connections (e.g. u_{11} and u_{12}) with input 1 and 2 respectively, and in total, there are $2 \times 25 \times 25$ weight connections. Fig 3.20 (b) shows the associated initial weight data. As can be seen, the initial weight values are evenly distributed.



(a) (b)

Figure 3.20: The KNN weight distribution map

(a: A two-input 25x25 KNN architecture; b: The initial weight distribution map (x-axis: the values of the weights associated with first neuron; y-axis: the values of the weights associated with second neuron)

Fig 3.20 (c) shows the weight distribution after applying three applications, each of them with two inputs (R1 and R4). As can be seen, four patterns have been produced in this test.

By using a statistical technique such as Linear Discriminant Analysis (LDA), the question how to relate four patterns to three applications can be resolved[13]. However, when the input number is larger than 2, the weight distribution becomes a high-dimensional like 3-D or 4-D structure, which imposes complexity to link the responded patterns with relevant traffic classes.

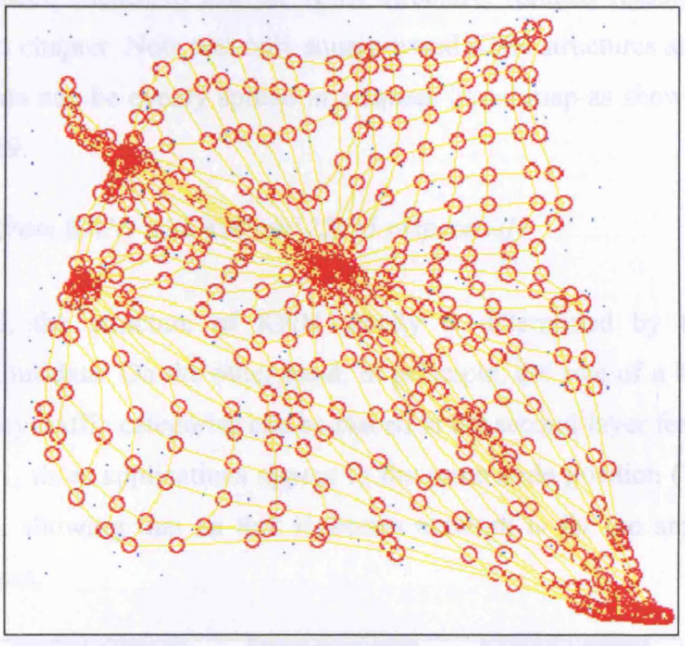


Figure 3.20 (c): The snapshot of the weight distribution map of a 25x25 KNN

3.7.1.2 KNN's output layer graph and IP flow profile map results

As discussed above, it is not feasible to use the weight distribution graph as the flow profile map when the input number exceeds two. Fortunately, there exists another option, which uses the output layer graph of KNNs as the IP flow map, as its dimension is always 2-D regardless of the input number.

[13] This is because the application number is three.

It is apparent that the structure of a KNN is important for traffic classification. Here, the structure mainly refers to the output layer size. If the structure is small, overlaps may occur and the overlarge structure causes the waste of computation resource. The following section presents a series of tests with different KNN structures. Two assumptions are made. One is that the input number for KNN is three. The other is that the input set of {R1,R2,R4}[14] has been chosen to test the KNN structure. Related reasons for that are discussed in the next chapter. Note that only square-based KNN structures are being tested as the converged data can be evenly spread in a square-based map as shown in Fig 3.17, Fig 3.18 and Fig 3.19.

- *Testing results from KNNs with 8x8 and 25x25 respectively*

One the one hand, the structure of KNN mainly is determined by the nature of characteristics of input data. On the other hand, in principle, the size of a KNN structure determines how many traffic categories can be placed in the second-layer feature map. For instance, in Fig 3.21, three applications appear in the same node position (7,0) in an 8x8 KNN after training, showing that an 8x8 Kohonen network is far too small for traffic classification purposes.

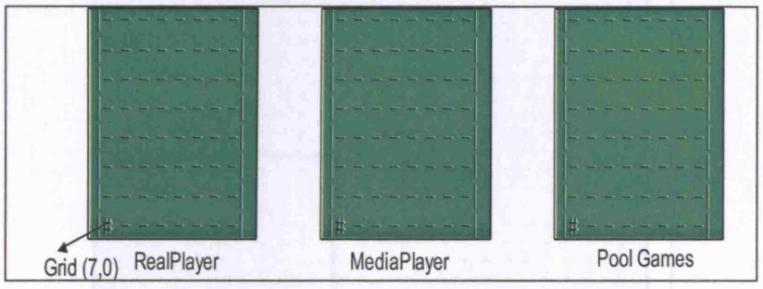


Figure 3.21: Testing results with an 8x8 Kohonen net after 500 training steps

A KNN with the 25x25 structure has a larger space than the previous one. Fig 3.22 is a snapshot of the first weight plane (because the input number is three). It shows that some converged areas (marked by boxes) have been produced on this type of KNN. However, overlaps occur in some areas. For instance, the weight data ranging from 0.21 and 0.28

[14] As will be discussed in Chapter 4, the input set of {R1,R2,R4} allows the proposed classifier to produce tight traffic patterns.

appear in varying areas. This phenomenon certainly affects the classification results as shown in Fig 3.23, where although Pool Games and MediaPlayer have already been clustered in separated areas; Realplayer has not since it spans two different places.

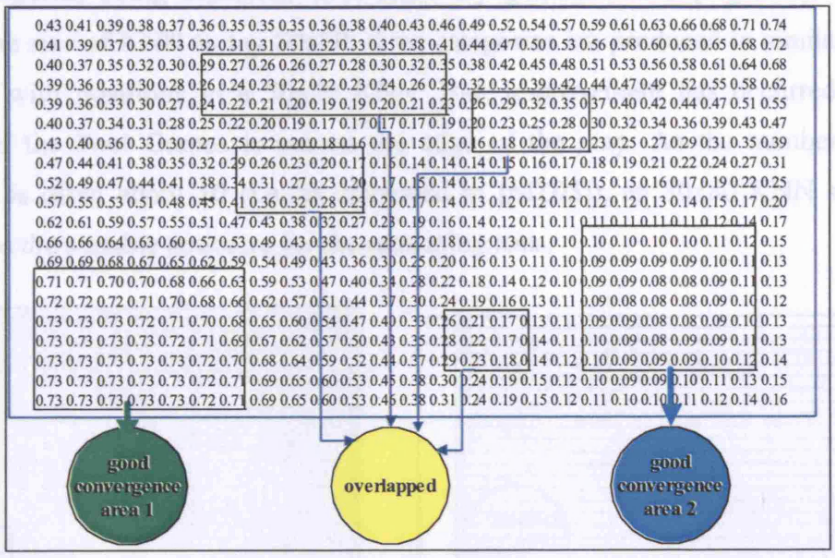


Figure 3.22: The weight values on the first plane of the 25x25 Kohonen network.

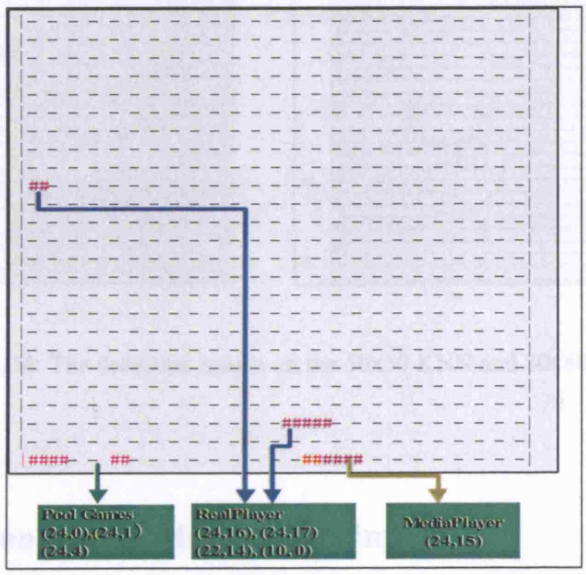


Figure 3.23: Testing results with a 25x25 Kohonen network

- *Testing results on 50x50 and 80x80 KNNs*

The testing outcomes on a 50x50 map certainly are better than previous results, three traffic categories being separated with relatively good distance (Fig.3.24). To further increase the size of KNN to say 80x80, three categories are produced in similar positions compared with positions in a 50x50 KNN. But, improvement has occurred since the position of the Pool Games is not on the edge of the map. As the number of traffic categories is often about 10 (i.e. as discussed in [MZ05]), an 80x80 KNN structure is regarded as the suitable structure for the remaining tests.

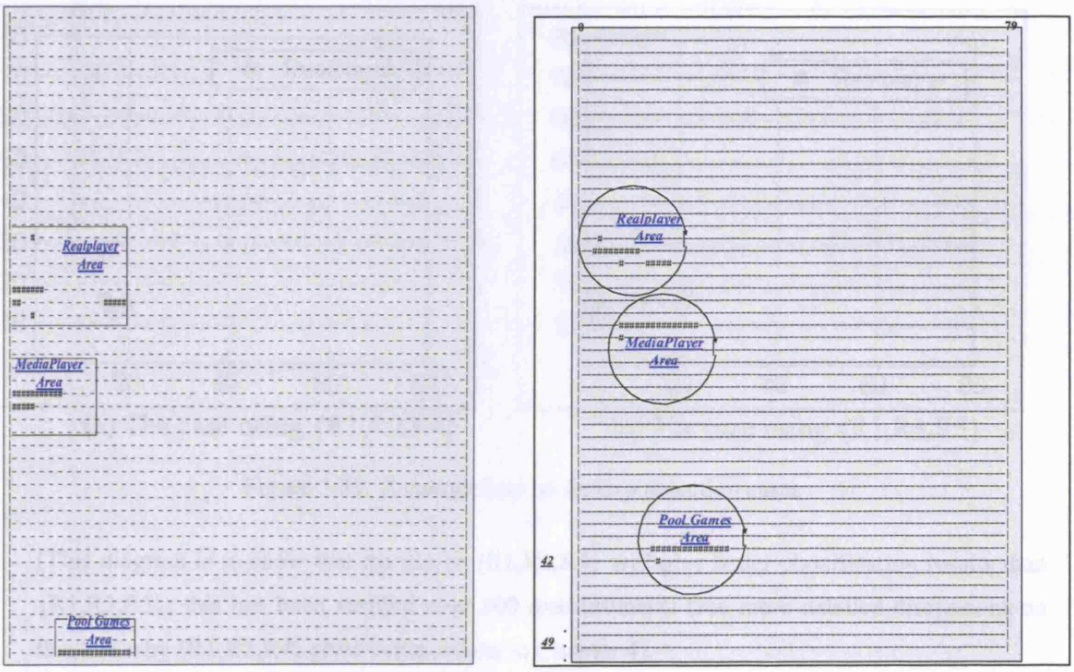
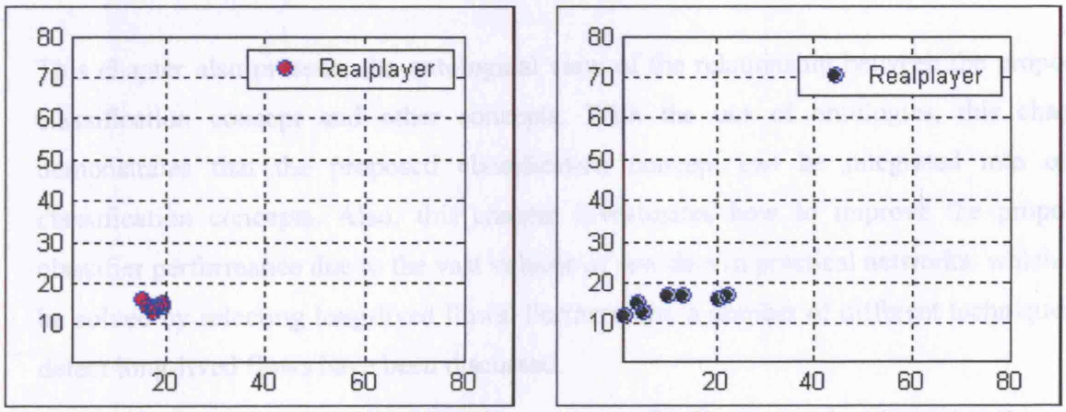


Figure 3.24: The detection results on the 50x50 KNN and 80x80 KNN

3.7.2 Classification results with random inputs

To show the effect of KNN structure on the yielded classification results, the tests shown above only use a specific parameter set: R1, R2 and R4. The reasons for choosing such a

set of parameters will be discussed in Chapter 4. However, whether the classification outcomes are accurate or not remains unknown; similarly, there is the question whether to use random set of inputs for classification. A comparison of the following two diagrams demonstrates such an issue when two different input sets of Realplayer are chosen: Fig 3.25(a) is the result of using {R1,R2,R4} while Fig 3.25 (b) is the result of using {R1,R3,R5}. Performed in the same testing environments, as it can be seen, the second detected pattern is more scattered than the first pattern, resulting in the possibility of greater inaccuracy compared with the use of {R1,R2,R4}. This suggests that data pre-processing also plays an important role in KNNs.



(a) The case using {R1,R2,R4} (b) The case using {R1,R3,R5}

Figure 3.25: A comparison of the two detection cases

(This diagram is to show that the use of {R1,R2,R4} will give better classification results than {R1,R3,R5}; this has been verified over 100 measurements (see more detailed discussions on this and why (R1,R2,R4) gives better results in Chapter 4).

3.8 Chapter summary

This chapter firstly discusses two concepts used for classification. One is the context concept while the other is the IP flow profile concept. The use of the context concept allows all the information related to classification to be used in an organised way and

hence minimises the effect due to the boundaries from different layers of input information, where current terms like ports or network diameters can not solve for they are too narrow and specific. The flow profile concept is defined to aid organisation of the input contexts at flow-level and to aid the purpose of automatic traffic detection. The flow profile of a network application is robust and stays unchanged no matter whether the application traffic has been wrapped into other protocols or has been encrypted. In addition, five contexts that could be used as the core inputs to model flow profiles are proposed, which only requires IP header information that is easy and fast to collect.

This chapter also presents the ontological view of the relationship between the proposed classification concept and other concepts. With the use of ontologies, this chapter demonstrates that the proposed classification concept can be integrated into other classification concepts. Also, this chapter investigates how to improve the proposed classifier performance due to the vast volume of raw data in practical networks: which can be solved by selecting long-lived flows. Furthermore, a number of different techniques to detect long-lived flows have been discussed.

Following the discussion of the relationship between an IP flow profile and its traffic class, this chapter examines the reasons for using KNN for classification. This is because the Gaussian based learning algorithms match the statistical nature of the input data, whose distributions generally can be modelled as Gaussian. What is more, KNN is able to automatically transform a high-dimensional input space into a low-dimensional map.

Methodologies to transform raw data from the raw traffic collection, to the construction of input contexts and to the realisation of KNN's using Matlab have been presented in this chapter.

Finally, the chapter presents a series of proof-of-concept experiments. The testing results demonstrate the powerful classification ability of KNNs, for the selected three applications have been well separated in different areas in an 80x80 structure. Even applications holding analogous traffic profiles like Realplayer and Mediaplayer have been effectively separated by KNNs. However, the detection results are based on the assumption that the input contexts are R1, R2 and R4. When the classification is made with random input selection, the experimental results show that the detected patterns may be more scattered than the results arising from using R1, R2 and R4. This raises the issue that the input selection is an important factor in terms of the classification accuracy. This issue will be addressed in the next chapter.

Chapter 4

Towards Accurate Traffic Classification

4.1 Introduction

The last chapter focuses on building an automatic traffic classifier through the use of KNNs. The detection results for three applications show that the proposed scheme is able to automatically recognise different traffic classes, transforming them into different patterns in a 2-D KNN feature map. Correspondingly, the classification accuracy largely lies in the shape of the detected patterns. The less scatter within a pattern and the more distance between patterns, the better detection and accuracy will be achieved.

Yet, how to obtain accurate classification results still remains to be answered. This chapter goes further to address a number of key issues in the realisation of which will lead to improved classification accuracy. First of all, as mentioned earlier, the input selection (also called the feature selection) plays an important role in classification in terms of the tightness of the detected patterns. Secondly, the detected patterns may overlap each other depending on the nature of the detected network traffic. Thirdly, as the shape of the traffic patterns produced by KNNs is often irregular, how to relate patterns to traffic classes may be an issue if the detected patterns are placed too close to each other. This chapter is structured as follows. Section 2 discusses how Principal Component Analysis is a useful technique in feature selection and followed by section 3 which examines the overlap problems and proposes Linear Discriminant Analysis to deal with the overlap problem. Section 4 and 5 present the proposed classification architecture and some aspects

regarding implementations of PCA and LDA respectively. Section 6 evaluates how PCA and LDA could help improve traffic classification results, and describes a series of test experiments to evaluate the value of the approach for five of the most common and current traffic classes.

4.2 Feature Selection

Features are defined as the useful attributes or primitives for pattern characterisation [JOA01]. Feature selection is an essential step in data pre-processing for a learning system [PEC04]. Appropriate selection minimises the redundant or irrelevant information on the one hand, while choosing the most representative features for classification on the other hand. Feature selection can be realised through either filter or wrapper methods as stated in [MZ05]:

> *Filter methods use the characteristics of the training data to determine the relevance and importance of certain discriminators to the classification problem. One the other hand, wrapper methods make use of results of a particular classifier to build the optimal set by evaluating the results for the classifier on a test set of different combinations of discriminators.*

This section discusses the use of PCA, a filter approach for the purpose of feature selection.

4.2.1 Principal Component Analysis based Input Selection for KNN

4.2.1.1 Basic idea

When the dimension of a feature space is low, feature selection can be realised through manual analysis of the correlations of inputs [MZ05]. The correlations are typically

expressed by covariance or correlation coefficient values. For example, six correlation coefficient values can be easily calculated for the three-input case. Given any n-dimensional input space, there will be $\frac{n!}{(n-2)!*2}$ correlation coefficient results and the correlation coefficient matrix will be:

$$C_X = \begin{bmatrix} cor(c_1,c_1) & cor(c_1,c_2) & \dots & cor(c_1,c_n) \\ cor(c_2,c_1) & cor(c_2,c_2) & \dots & cor(c_2,c_n) \\ \dots & \dots & \dots & \dots \\ cor(c_n,c_1) & cor(c_n,c_2) & \dots & cor(c_n,c_n) \end{bmatrix} \quad (4.1)$$

Clearly, it is infeasible to analyse a large C_X manually. A more effective solution is to deploy PCA to transform a high-dimensional C_X into a low-dimensional matrix, while still preserving original data features as discussed below.

4.2.1.2 Principal Component Analysis

Principal component analysis (PCA) has been widely used for feature selection in the supervised learning system [PEC04][MG04][DIJ06]. But, to the author's knowledge, the application of PCA to traffic classification in the unsupervised learning system has not been reported. Instead of using original inputs that may highly correlate with each other, PCA provides fewer uncorrelated principal components (PC) for classification i.e. lower dimensional correlation matrix. Fig 4.1 illustrates the geometric meaning of PCA in a 2-D input space. Here the first PC (PC1) reflects the direction holding the first most variance of inputs, while the orthogonal direction PC2 is the second direction. In the n-dimensional input case (n>3), it is typically that the first 2 or 3 principal components encompass >85% of the total variance of inputs [WOL03]. Hence, through the use of PCA, it is possible to transform to fewer inputs for classification, rather than the most extensive original data.

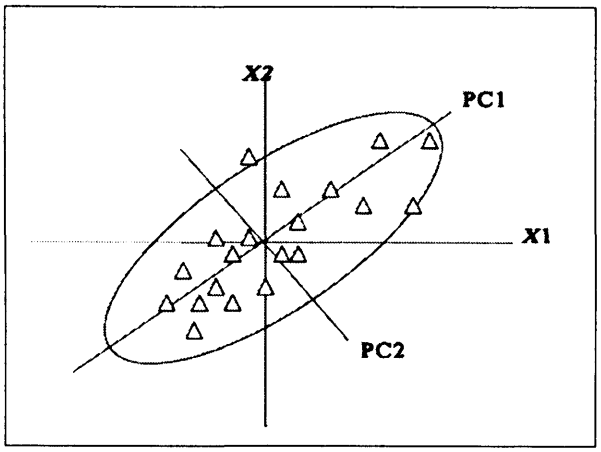


Figure 4.1: The geometric illustration of a 2-D PCA case

PCA algorithms

The reason why a much smaller number of PCs can represent the majority of input data features is explained as follows.

As C_X is the square symmetric matrix (where $x_{ij} = x_{ji}$), it guarantees that the corresponding orthogonal matrix C_Y exists [JAM85]:

$$C_Y = AC_X A^T \tag{4.2}$$

where A is the eigenvectors of C_X, and C_Y satisfies:

$$C_Y = \begin{bmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & \lambda_n \end{bmatrix}$$

where λ_k is the eigenvalue of C_X. Y is the principal component vector, a multiplication result of $[X - X_m]$ and A, where $X_m = E(X)$. In addition, as shown above in C_Y, since

the correlation between any two elements of Y is zero, Y is therefore an uncorrelated vector.

$$Y = A \times [X - X_m] \tag{4.3}$$

The elements of eigenvectors, also called factor loadings, determine the weights in terms of the contribution of original feature data to principal components. Given an eigenvector matrix is:

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ 0 & 0 & \dots & 0 \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix}$$

Y can be rewritten with an expanded linear form as follows (Note that both Y_1 and PC1 mean the first Principal Component.):

$$Y = \begin{cases} Y_1 = a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n \\ Y_2 = a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n \\ \dots \\ Y_3 = a_m x_1 + a_{,12}x_2 + \dots + a_{mn}x_n \end{cases} \tag{4.4}$$

Due to the fact that C_X is a square symmetric matrix, this guarantees that its eigenvalues and eigenvectors exist [JAM85]:

$$|C_X - \lambda I| \times A = 0 \tag{4.5}$$

This equation is a function of λ as shown below:

$$f(\lambda) = \lambda^n + p_1\lambda^{n-1} + p_2\lambda^{n-2} + \dots\dots + p_n = 0 \tag{4.6}$$

where p_i stands for the compound coefficient from matrix C_X. For example, if matrix $C_X = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix}$, then $f(\lambda) = \begin{bmatrix} c_{11} - \lambda & c_{12} \\ c_{21} & c_{22} - \lambda \end{bmatrix}$, which also equals:

$$f(\lambda) = (c_{22} - \lambda)(c_{11} - \lambda) - c_{12}c_{21} = \lambda^2 - (c_{22} + c_{11})\lambda + (c_{22}c_{11} - c_{12}c_{21}) \quad (4.7)$$

where $p_1 = -(c_{22} + c_{11})$ while $p_2 = c_{22}c_{11} - c_{12}c_{21}$

Having calculated λ, the corresponding eigenvector A can be gained using the Equation 4.3, as well as the uncorrelated vector Y. Notice that the first row[15] of A often corresponds to the largest value of λ then the second row of A related to the second largest value in λ and so on. Therefore, the first component in Y always holds the highest variability and then the second component and so on. Usually, the first two or three PCs hold over 85%-90% of the overall variance. Thus, PCA transforms the high multi-dimensional data into a small dimensional data space.

4.2.1.3 Interpretation of Principal Components

The use of PCA for feature selection depends on the requirements of the learning system. When it comes to finding the most representative input data to describe related statistical features (i.e. variability), a transformed vector Y can be directly used, which typically employs the first few principal components. This kind of selection is also called feature extraction as the selected inputs actually are not original data but the transformed principal components. This method has been widely adopted for feature selection or extraction in supervised learning system [PEC04][MG04][DIJ06]. Nevertheless, for the classification task of interest here, there is a potential problem that may result in situations where the chosen principal component corresponds to the attributes with the highest variability but with weak or without any discriminating power [DIJ06]. Instead, as stated [ED01], the selection of the last few principal components is often practically useful in many cases

[15] In matlab, it is the first column.

since, as a whole, they hold the least variance. Here shares the same idea as [ED01] as the less variability introduces the better pattern recognition results in Kohonen Neural Network.

However, as it is inappropriate to directly use the transformed PCs (namely, feature extraction), the thesis will examine the factor loadings (i.e. a_{ij} in Y) in terms of related contribution to PCs. More detailed investigation of the effect of factor loading can be found in section 4.6.1. However, an introduction of these roles can be seen from the following model example. In the case of a model involving a five-input flow profile, assuming that Y1 and Y2 (the first two PCs) hold over 85% of the total variability, then the input selection is realised according to the value of the factor loadings. In Equation 4.3 (also seen below), Y1 is a component where the main contribution to the variability comes from x_3 and x_5 due to the large factor loading values a_{13} and a_{15} respectively. Thus, a heuristic selection of x_1, x_2 and x_4 can be made since they contribute much less to the variability in Y1 in comparison to others. Similarly, in case of Y2, x_1, x_3 and x_4 are chosen. Overall, parameter x_1, x_2 and x_4 can be viewed as the optimal input set for classification for other two parameters contribute the most of variability to PCs as a whole. Although this selection criterion is a heuristic approach, the later experimental results illustrate that it is an efficient and reliable solution.

$$Y_1 = a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + a_{14}x_4 + a_{15}x_5$$
$$Y_2 = a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + a_{24}x_4 + a_{25}x_5$$

$$A_1 = \begin{bmatrix} 0.0160 & 0.1892 & 0.6389 & 0.3937 & 0.6330 \end{bmatrix}$$
$$A_2 = \begin{bmatrix} 0.0606 & 0.5107 & 0.1913 & 0.0598 & 0.6528 \end{bmatrix}$$

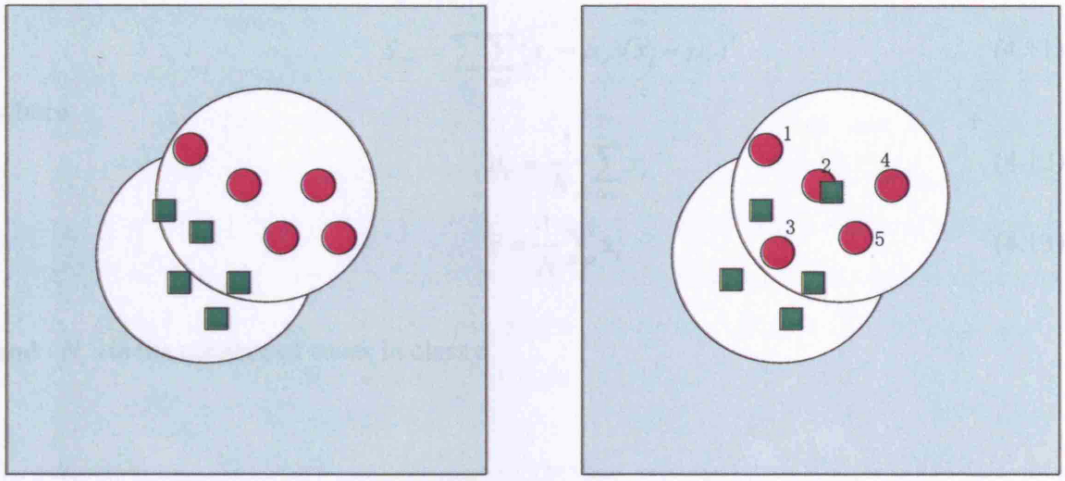
4.3 The overlap issues

4.3.1 Different overlap classes

Another major problem hampering traffic classification accuracy is the overlap issue between detected patterns. Overlaps arise because of small differences between some network applications. Overlaps can appear in different forms, but in general they can be grouped into the following two classes:

Partial overlap refers to detected patterns that are very close to each other in terms of their Euclidean distances. For example, the following diagram (Fig.4.2-a) shows two classes, each consisting of five responded neurons, labelled with pink and green respectively. They are partially overlapped if marked with the traditional means i.e. using circles. For example, the mean of responded neuron positions is regarded as the centre of the circle while the maximum distance from the neurons to the circle's centre equals the diameter of the circle.

Complete overlap means the case that one position like a neuron responds to over one traffic classes like neuron 2 in the following diagram.



(a) The partial overlap (b) the complete overlap

Figure 4.2: The illustration of two overlap cases

4.3.2 Linear Discriminant Analysis based solutions

The Linear Discriminant Analysis (LDA) technique is effective at dealing with the partial overlap problem. It is a mechanism trying to seek a linear data projection, which is:

$$y = v'X \quad \Rightarrow \quad y = v_1 x_1 + v_2 x_2 + ... + v_n x_n \tag{4.8}$$

while the vector v required to satisfy the condition such that the ratio between the between-classes variance to the within-class is maximised:

$$\lambda = \frac{v' S_B v}{v' S_W v} \tag{4.9}$$

where S_B is the 'between-class scatter matrix' and S_W stands for the 'within-class scatter matrix'. The calculations of the two scatter matrices are[MIK99]:

$$S_B = \sum_c (\mu_c - \bar{x})(\mu_c - \bar{x})^T) \tag{4.10}$$

$$S_W = \sum_c \sum_{i \in c} (x_i - \mu_c)(x_i - \mu_c)^T \tag{4.11}$$

where

$$\mu_c = \frac{1}{N_c} \sum_{i \in c} x_i \tag{4.12}$$

$$\bar{x} = \frac{1}{N} \sum_i x_i \tag{4.13}$$

and N_c is the number of cases in class c.

The problem of maximising λ can be regarded in the usual way as setting the differentiating equation $\frac{d\lambda}{dv}$ to zero [JAM85]:

$$\frac{d\lambda}{dv} = \frac{2[S_B v(v' S_W v) - (v' S_B v) S_W v]}{(v' S_W v)^2} = 0 \tag{4.14}$$

By dividing top and bottom with $v' S_W v$, Equation (4.14) is equal to:

$$\frac{2[S_B v - \lambda S_W v]}{v' S_W v} = 0 \tag{4.15}$$

Therefore, it follows from Equation 4.15 that:

$$S_B v = \lambda S_W v \tag{4.16}$$

If S_W is non-singular, S_W^{-1} exists, then Equation 4.16 is equivalent to:

$$S_W^{-1} S_B v = \lambda v \tag{4.17}$$

Thus, the computation of the maximum λ becomes a matter of finding the largest eigenvalue of the $S_W^{-1} S_B$ matrix.

For example, as shown in Fig 4.3, the projection to the lower right axis achieves the maximum separation between two classes while that to the lower left axis gives the worst separation. Therefore, when λ is maximised, the partial overlap can be effectively diminished to the lowest level on the new linear projection axis. The use of LDA also makes the reading of different traffic classes easier. The projection value of the mean of the projected data is often regarded as the separation value, which is used to distinguish the two projected classes. In this thesis, such a separation value is called the separation point e.g. the black small box shown in the right projection line in Fig 4.3.

This thesis focuses on the linear projection approach for reasons of simplicity, as well as gaining advantage from the large amount of prior work in this area [DIJ06]. However, the potential for non-linear projection will be discussed in the future section 7.x y.

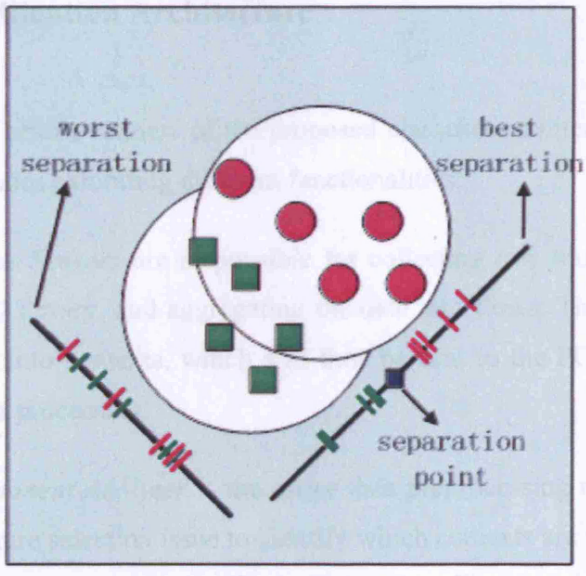


Figure 4.3: The comparison between the best and the worst project lines.

4.3.3 Using Alternative Maps

Apparently, the complete overlap problem is even worse and harder to solve than the partial overlap issue and in general it can not be solved through the LDA technique. Since the complete overlap problem is caused by the natural characteristics of the input data, one of the possible solutions is to construct another alternative map by selecting different set of inputs. Obviously, different patterns will be produced when input parameters change and as such complete overlap may be avoided. This process can be repeated until a good pattern is found. This is a typical *wrapper* method as discussed in section 4.2.

4.4 Traffic Classification Architecture

The section gives a brief overview of the proposed classifier architecture, which consists of a number of modules exhibiting different functionalities:

1. *Network Context Sensors* are responsible for collecting raw traffic using the *packet capture (PCAP)* library, and aggregating the data into flows. Then, the raw data can be transformed into contexts, which will then be sent to the PCA and KNN for the next step in data processing.

2. *Principal Component Analyser* is the major data preprocessing module. It deals with the optimal feature selection issue to identify which contexts are suitable for the KNN learning stage. This module is mainly applied in the offline training stage.

3. *Kohonen Neural Network Analyser* (KNNA) performs the nonlinear learning and provides mapping functionalities for IP flow profiles. After successful training, the structural data like input vector or weight matrix is stored in the KNN repository. During the online testing, KNNA will automatically load the needed data from this repository. Through the proper training and learning, KNNA provides the position

data on different traffic categories and passes these to the Automated Traffic Classifier module.

4. *Automated Traffic Classifier (ATC)* carries out the automatic detection function. It uses the data from KNNA to yield an IP flow profile map (IFPM). It records relevant pattern position data for various traffic classes in an IFPM during the offline learning period. In the case that the received data overlaps with others, *ATC* will pass data to LDA for further offline processing.

5. *Linear Discriminant Analyser* deals with the overlap issue when some classifier patterns in the IFPM are closely located.

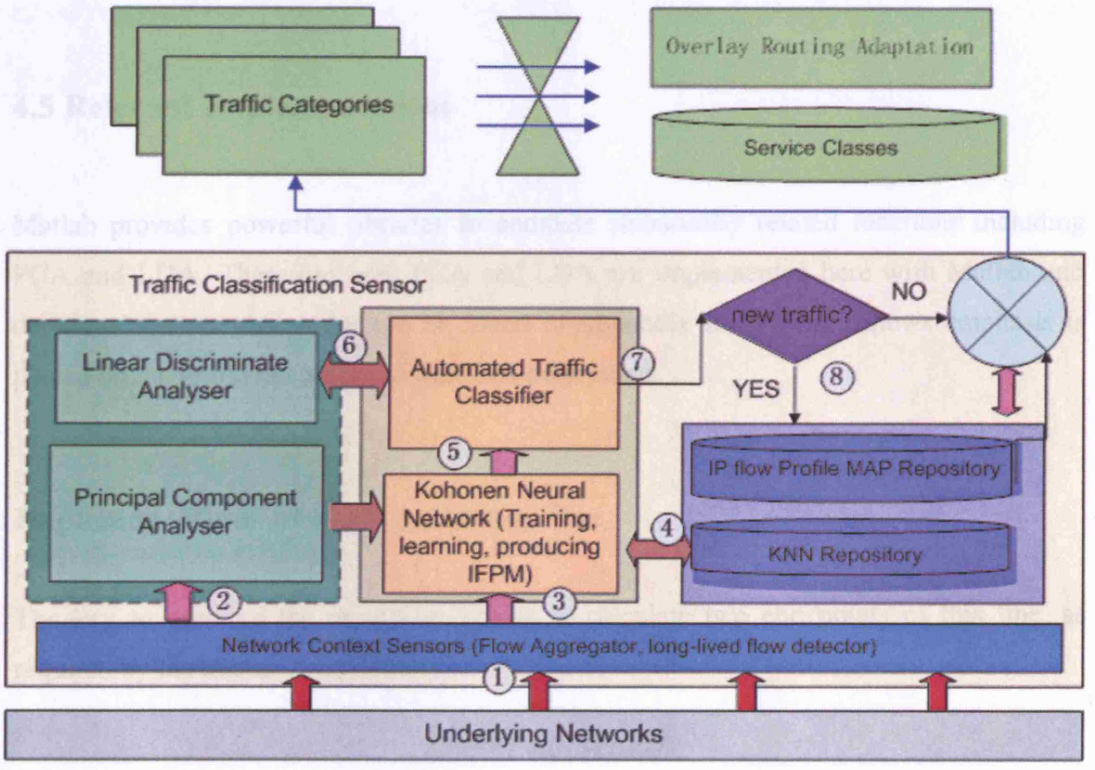


Figure 4.4: Automatic traffic classification sensor and its architecture

Fig 4.4 lists the 8 steps in the classification procedure. The raw data processing is performed in the 1st step and the results are sent to PCA and KNN for feature selection and

producing IFPM; these are performed in step 2 and step 3 respectively. Then, in step 4, the static structure data from KNN and IFPM is stored in related databases. In steps 5 and 6, the accuracy of results is further improved using LDA and this is followed by steps 7 and 8, where the detected traffic patterns are compared with registered IFPM and the classification results are produced.

Considering the background of how to deploy such traffic classification functionalities into real networks, a new term called traffic classification sensor is defined to represent the whole detection task. On top of the traffic classification sensors, other services can be derived like service class mapping (according to traffic categories), overlay routing adaptation etc.

4.5 Relevant Implementations

Matlab provides powerful libraries to compute statistically related functions including PCA and LDA. Therefore both PCA and LDA are implemented here with Matlab and details of the relevant codes can be found in Appendix B. In what follows emphasis is placed on the implementations of the LDA projections.

Implementations of LDA projection line

The key to drawing the projection line is to calculate two end points of this line, as required by the matlab *line( )* function:

line([x1 x2], [y1 y2])

Let v be the eigenvector of the projection matrix and let (x_m, y_m) be the projection point of the mean value of inputs from the detected classes and the map size is 80x80.

There exist two different cases of the projection line drawing, depending on whether the line is more vertical or horizontal.

Case 1: When the project line is more horizontal, it means that x1=0 and x2=49 while y1 and y2 are unknown. Note that the eigenvector v geometrically means the direction of the projection line. Therefore, $v'(1,1)$ is the standardised value in x-axis direction while $v'(1,2)$ represents that in y-axis direction[16]. Let θ be the angle between the projection line to x-axis. Let y_{gap1} and y_{gap2} stand for the vertical distance from the mean point to (x1,y1) and (x2,y2) respectively. Note that both y_1 and y_2 could be positive or negative as shown in Fig 4.5 (a) and (b). If y_{gap1} and y_{gap2} are known, y_1 and y_2 can be resolved easily as shown below.

For y_1*:*

$$\because \quad \tan(\theta) = \frac{v'(1,2)}{v'(1,1)} = \frac{y_{gap1}}{x_m} \;;$$

$$\therefore \quad y_{gap1} = x_m \times \frac{v'(1,2)}{v'(1,1)} \;;$$

$$\therefore \quad y_1 = y_m - y_{gap1} = y_m - x_m \times \frac{v'(1,2)}{v'(1,1)} \;;$$

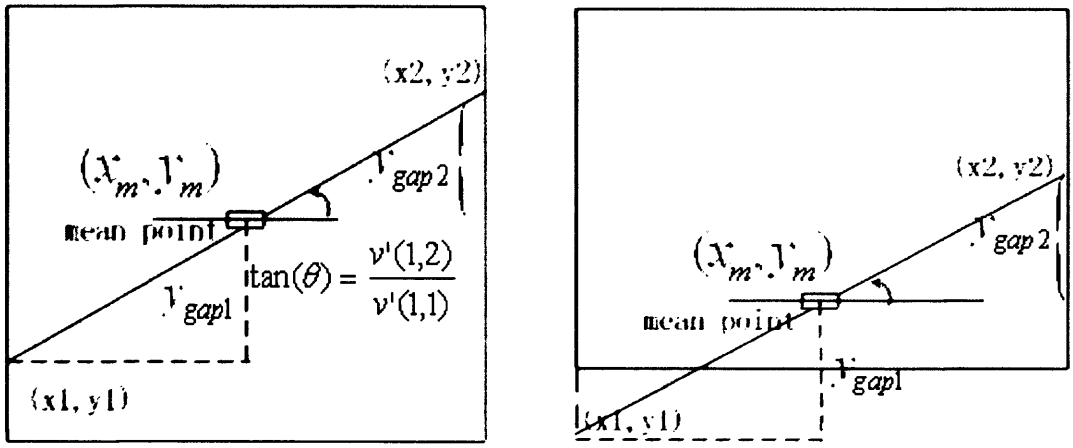
For y_2*:*

$$\because \quad \tan(\theta) = \frac{v'(1,2)}{v'(1,1)} = \frac{y_{gap2}}{x_2 - x_m} \;;$$

$$\therefore \quad y_{gap2} = (x_2 - x_m) \times \frac{v'(1,2)}{v'(1,1)} \;;$$

$$\therefore \quad y_2 = y_m + y_{gap2} = y_m + (x_2 - x_m) \times \frac{v'(1,2)}{v'(1,1)} \;;$$

[16] As the results of eigenvector max produced by Matlab is column based, v needs to be reserved to row based (v').



(a) $y_1>0$ (left); (b) $y_1<0$ (right)

Figure 4.5: The horizontal case

Case 2: In the case that the project line is more vertical, y1=0 and y2=49 while x1 and x2 are unknown. Let x_{gap1} and x_{gap2} stand for the horizontal distance from the mean point to (x1,y1) and (x2,y2) respectively and both x_1 and x_2 could be positive or negative as shown in Fig 4.6. x1 and x2 can be resolved as follows:

For x_1*:* $\because$ $\quad \text{ctan}(\theta) = \dfrac{v'(1,1)}{v'(1,2)} = \dfrac{x_{gap1}}{y_m}$;

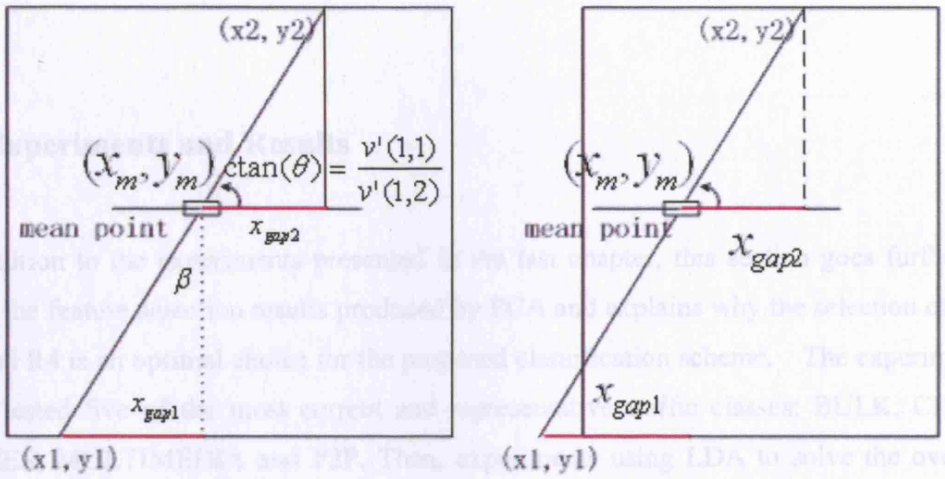
$\therefore$ $\quad x_{gap1} = y_m \times \dfrac{v'(1,1)}{v'(1,2)}$;

$\therefore$ $\quad x_1 = x_m - x_{gap1} = x_m - y_m \times \dfrac{v'(1,1)}{v'(1,2)}$;

For x_2*:* $\because$ $\quad \text{ctan}(\theta) = \dfrac{v'(1,1)}{v'(1,2)} = \dfrac{x_{gap2}}{y_2 - y_m}$;

$\therefore$ $\quad x_{gap2} = (y_2 - y_m) \times \dfrac{v'(1,1)}{v'(1,2)}$;

$\therefore$ $\quad x_2 = x_m + x_{gap} = x_m + (y_2 - y_m) \times \dfrac{v'(1,1)}{v'(1,2)}$;



(a) $x_1>0$ (left) (b) $x_1<0$ (right)

Figure 4.6: The vertical case

4.6 Experiments and Results

In addition to the experiments presented in the last chapter, this section goes further to show the feature selection results produced by PCA and explains why the selection of R1, R2 and R4 is an optimal choice for the proposed classification scheme. The experiments have tested five of the most current and representative traffic classes: BULK, CHAT, GAMES, MULTIMEDIA and P2P. Then, experiments using LDA to solve the overlap problems are demonstrated.

4.6.1 Feature Selection Results

4.6.1.1 Initial selections

As shown in Table 4.1, nine inputs are available for classification. As discussed in section 3.2.3 and 3.6.1, the first three contexts (C1, C2, C3) are regarded as primitive data while the last two contexts (C4, C5) may exhibit some degree of correlations with previous three contexts. This can be found out through examination of the correlation coefficient matrix. As shown in table 4.2, the correlation among R2, R6 and R8 is very high (>0.9); the correlation between R3, R6 and R8 are moderate; the correlation between R5, R7 and R9 are moderate while the correlation among R5, R7 and R9 is also showing high (negative 0.7). Such correlation results reflect the highly correlated relationship between inputs R2-5 and inputs R6-9. Therefore, to reduce the redundancy, we can choose the input set to be either {R1,R2,R3,R4,R5} or { R5,R6,R7,R8,R9} for traffic classification. Note that the computation of from R6 to R9 requires the knowledge of link bandwidth in advance, which introduces some difficulty accordingly. By contrast, it is rather easy to compute R1-4 data. Hence, the input set {R1, R2, R3, R4, R5} is chosen for the rest of testing.

Table 4.1: Skype's input data

R1	R2	R3	R4	R5	R6	R7	R8	R9
0. 0387	0. 0987	0. 0394	0. 0866	0. 475000	0. 000047	0. 000054	0. 000047	0. 000055
0. 0403	0. 0196	0. 0406	0. 0198	0. 500000	0. 000247	0. 000244	0. 000247	0. 000246
0. 0404	0. 0196	0. 0408	0. 0196	0. 500000	0. 000247	0. 000247	0. 000247	0. 000249
0. 0406	0. 0195	0. 0408	0. 0218	0. 530000	0. 000250	0. 000224	0. 000250	0. 000225
0. 0413	0. 0197	0. 0382	0. 0964	0. 845000	0. 000252	0. 000051	0. 000252	0. 000048
0. 0413	0. 0196	0. 0393	0. 0607	0. 735000	0. 000253	0. 000082	0. 000253	0. 000078
0. 0413	0. 0195	0. 0390	0. 0196	0. 500000	0. 000253	0. 000253	0. 000253	0. 000239
0. 0409	0. 0196	0. 0360	0. 0665	0. 780000	0. 000250	0. 000074	0. 000250	0. 000065
0. 0381	0. 0197	0. 0375	0. 0214	0. 530000	0. 000232	0. 000213	0. 000232	0. 000210
0. 0377	0. 0196	0. 0339	0. 1236	0. 865000	0. 000231	0. 000037	0. 000231	0. 000033
0. 0376	0. 0196	0. 0371	0. 0401	0. 650000	0. 000231	0. 000113	0. 000231	0. 000111
0. 0377	0. 0197	0. 0375	0. 0235	0. 545000	0. 000230	0. 000193	0. 000230	0. 000192
0. 0376	0. 0194	0. 0374	0. 0286	0. 645000	0. 000232	0. 000158	0. 000232	0. 000157
0. 0377	0. 0196	0. 0377	0. 0255	0. 510000	0. 000231	0. 000177	0. 000231	0. 000177
0. 0378	0. 0198	0. 0368	0. 0335	0. 690000	0. 000229	0. 000135	0. 000229	0. 000132
0. 0376	0. 0194	0. 0375	0. 0299	0. 585000	0. 000232	0. 000151	0. 000232	0. 000150
0. 0377	0. 0196	0. 0372	0. 0279	0. 540000	0. 000231	0. 000162	0. 000231	0. 000160
0. 0377	0. 0196	0. 0376	0. 0227	0. 535000	0. 000231	0. 000199	0. 000231	0. 000199
0. 0377	0. 0197	0. 0374	0. 0338	0. 635000	0. 000231	0. 000134	0. 000231	0. 000133
0. 0376	0. 0196	0. 0375	0. 0265	0. 575000	0. 000231	0. 000170	0. 000231	0. 000170

Table 4.2: The correlation coefficients

	R1	R2	R3	R4	R5	R6	R7	R8	R9
R1	1. 0000								
R2	-0. 0165	1. 0000							
R3	-0. 0087	0. 0590	1. 0000						
R4	-0. 4475	-0. 0141	-0. 0141	1. 0000					
R5	0. 8059	-0. 0646	0. 0081	-0. 6590	1. 0000				
R6	-0. 0317	0. 9420	-0. 2521	-0. 0019	-0. 0753	1. 0000			
R7	-0. 9721	0. 0276	0. 0148	0. 3719	-0. 7197	0. 0411	1. 0000		
R8	-0. 0316	0. 9421	-0. 2520	-0. 0019	-0. 0754	1. 0000	0. 0410	1. 0000	
R9	-0. 9509	-0. 0003	0. 0174	0. 5250	-0. 7142	0. 0142	0. 9734	0. 0142	1. 0000

4.6.1.2 Selection Results of PCA

This section presents the PCA selection results on five traffic categories: BULK (*ftp*-data), P2P (*bt*-data), Multimedia (Realplayer-data), Chatting (Skype-voice) and Games (pool). All the traces on five categories do not contain any control/protocol data but only IP header information.

As discussed in section 4.2, what PCA basically does is to project original data into a new set of principal components (PCs) in descending order according to their relative variability. Such PCs are the feature extraction from the original statistically data and can not be directly used for the traffic classification task, as they only represent the variability feature rather than the discriminatory characteristics. For the classification task, the less variability exhibited by the input data set, the better, as tighter KNN patterns will result. Thus, the feature selection task aims to find out which inputs contribute less variability to principal components as explained below.

Table 4.3 shows sixteen groups of the original data from Realplayer traffic. Table 4.4 gives the related eigenvectors, the eigenvalues and shows that the first two components accounts for 83.56% of the whole variability. According to Equation (4.4), the principal components are computed as follows[17]:

PC1= -0. 0470*X1-0. 5899*X2+0. 5411*X3+0. 3595*X4+0. 4773*X5;
PC2= 0. 6852*X1+0. 2030*X2-0. 3719*X3+0. 3883*X4+0. 4475*X5;

From the above equations, it can be seen that the least loading factors to PC1 are {input 1, input 4} while that to PC2 are {input 2, input 3, input 4}. Similarly, for Skype traffic, the least loading factors to PC1 are {input 2, input 3} while that to PC2 are {input 1, input 4, input 5}. The least loading factors to PC1 and PC2 are {input 1, input 4, input 2} and {input 2, input 5} for P2P traffic, respectively while those least loading factors to FTP's PC1 and PC2 are {input 1, input 2, input 4} and {input 3, input 5} respectively. Also, the least loading factors to Pool's PC1 and PC2 are {input 1} and {input 4} respectively. Table 4.9 gives a summary regarding contribution of inputs made to related PCs. This Table shows that input 1, 4 and 2 are ranked as the inputs that contribute the least to related PCs. According to this result, parameters R1, R2 and R4 is the feature selection result for traffic classification. It is clear that the input number is not limited to three. The reason to limiting to three inputs is because the larger the number of inputs, the greater are the correlation possibilities among data; extra inputs also introduce extra overhead. In

[17] Let us assume that X_i=[X-Xm].

addition, as shown in Table 4.9, it is hard to choose which data should be the fourth input since the ranking of the fourth input is the same as the fifth input.

Table 4.3: Realplayer's inputs: R1-R5

#	R1	R2	R3	R4	R5
1	0. 024	0. 073778	0. 28729	0. 33667	0. 007779
2	0. 032	0. 161	0. 24381	0. 34709	0. 0079593
3	0. 008	0. 072667	0. 96915	0. 34409	0. 0078057
4	0. 008	0. 072667	0. 98385	0. 40684	0. 0095382
5	0. 012	0. 072667	0. 97465	0. 43978	0. 010345
6	0. 008	0. 059667	0. 969	0. 36833	0. 0085696
7	0. 008	0. 072667	0. 96823	0. 36388	0. 0085768
8	0. 008	0. 072667	0. 96927	0. 35755	0. 0082843
9	0. 008	0. 072667	0. 95957	0. 3422	0. 0078913
10	0. 008	0. 072667	0. 96955	0. 35748	0. 0082933
11	0. 008	0. 072667	0. 97683	0. 3329	0. 0077475
12	0. 008	0. 072667	0. 97045	0. 29988	0. 0069378
13	0. 008	0. 072667	0. 96564	0. 34431	0. 0079404
14	0. 02	0. 092533	0. 58796	0. 34085	0. 010574
15	0. 016	0. 066167	0. 96403	0. 34683	0. 017048
16	0. 02	0. 062267	0. 9661	0. 3717	0. 018533

Table 4.4: Eigenvectors and Eigenvalues from Realplayer's data

	v1	v2	v3	v4	v5
A11	-0. 0170	0. 6852	0. 3314	-0. 0154	0. 6467
A12	-0. 5899	0. 2030	-0. 2748	0. 7248	-0. 0998
A13	0. 5411	-0. 3719	-0. 0102	0. 6030	0. 4530
A14	0. 3595	0. 3883	-0. 8378	-0. 1281	0. 0410
A15	0. 4773	0. 4475	0. 3357	0. 3072	-0. 6041
λ	2. 2407	1. 9373	0. 5916	0. 1813	0. 0491
ratio	0. 4481	0. 3875	0. 1183	0. 0363	0. 0098
cdf	0. 4481	0. 8356	0. 9539	0. 9902	1
Var(PC)	2. 2407	1. 9373	0. 5916	0. 1813	0. 0491

Table 4.5: Eigenvectors and Eigenvalues from Skype's data

	v1	v2	v3	v4	v5
A11	-0. 5752	-0. 0217	-0. 0311	0. 5874	-0. 5680
A12	0. 0487	0. 7037	-0. 7030	0. 0794	0. 0443
A13	-0. 0058	0. 7053	0. 7077	0. 0329	-0. 0259
A14	0. 5210	-0. 0833	0. 0610	0. 7946	0. 2939
A15	-0. 6286	-0. 0010	0. 0180	0. 1270	0. 7670
λ	2. 2897	1. 0886	0. 9140	0. 5619	0. 1458
ratio	0. 4579	0. 2177	0. 1828	0. 1124	0. 0292

cdf	0. 4481	0. 8356	0. 9539	0. 9902	1
Var(PC)	2. 2897	1. 0886	0. 9140	0. 5619	0. 1458

Table 4.6: Eigenvectors and Eigenvalues from P2P's data

	v1	v2	v3	v4	v5
A11	0. 4166	-0. 5570	-0. 3172	0. 6214	-0. 1714
A12	0. 4371	0. 4099	-0. 7440	-0. 2914	0. 0506
A13	0. 4591	0. 4311	0. 4186	0. 1146	-0. 6442
A14	-0. 4185	0. 5531	-0. 2064	0. 6876	0. 0601
A15	0. 4994	0. 1731	0. 3579	0. 2074	0. 7413
λ	3. 7201	0. 8946	0. 2300	0. 1373	0. 0181
ratio	0. 7440	0. 1789	0. 0460	0. 0275	0. 0036
cdf	0. 7440	0. 9229	0. 9689	0. 9964	1
Var(PC)	3. 7201	0. 8946	0. 2300	0. 1373	0. 0181

Table 4.7: Eigenvectors and Eigenvalues from *ftp's* data

	v1	v2	v3	v4	v5
A11	-0. 0160	-0. 6061	0. 7814	0. 1338	-0. 0618
A12	0. 1892	0. 6107	0. 5672	-0. 5191	0. 0087
A13	0. 6389	-0. 1913	-0. 1604	-0. 1792	-0. 7052
A14	-0. 3937	-0. 3989	-0. 1792	-0. 8086	-0. 0022
A15	0. 6330	-0. 2528	-0. 0993	-0. 1634	0. 7062
λ	2. 2289	1. 2438	0. 8691	0. 6558	0. 0024
ratio	0. 4458	0. 2488	0. 1738	0. 1312	0. 0005
cdf	0. 4458	0. 6946	0. 8684	0. 9996	1
Var(PC)	2. 2289	1. 2438	0. 8691	0. 6558	0. 0024

Table 4.8: Eigenvectors and Eigenvalues from pool's data

	v1	v2	v3	v4	v5
A11	-0. 0058	0. 4623	-0. 8459	-0. 2660	-0. 0025
A12	-0. 4685	0. 5143	0. 1456	0. 4462	-0. 5437
A13	0. 5227	0. 3650	0. 3613	-0. 5218	-0. 4367
A14	-0. 5353	0. 3524	0. 3428	-0. 4707	0. 5001
A15	0. 4698	0. 5141	0. 1234	0. 4860	0. 5134
λ	2. 2289	1. 2438	0. 8691	0. 6558	0. 0024
ratio	0. 3988	0. 3085	0. 1813	0. 0615	0. 0498
cdf	0. 3988	0. 7073	0. 8886	0. 9501	1
Var(PC)	2. 2289	1. 2438	0. 8691	0. 6558	0. 0024

Table 4.9: Summary of each input's contribution to PCs (weight)

	contributions to PC1	contributions to PC2
Reaplayer	1 4	2 3 4
SKYPE	2 3	1 4 5
P2P	1 4 2	2 5
FTP	1 2 4	3 5
Pool	1	4
Total	input 1: (23.9%); input 4: (23.9%); input 2: (19.1%); input 3: (14%); input 5: (14%)	

4.6.1.3 Classification Results using parameters R1, R2 and R4

The classification outcomes for five traffic classes using R1, R2 and R4 are shown in Fig 4.7. First of all, the produced Skype pattern is not as tight as Realplayer's since Realplayer's packet size distribution is Gaussian whereas Skype's deviates from Gaussian (see 3.5.2). Secondly, as can be seen, the pattern of CHAT stays close to GAMES's, since there is a degree of similarity between their flow profiles. For example, both of them are symmetric and have *fpbd* values close to 0.5. Similarly, the distance between BULK's and P2P's patterns is relatively small, since they are both used for transferring files, which implies similar traffic profiles. In addition, it can be more difficult to distinguish closely spaced patterns and therefore it can be more difficult to relate them to their traffic classes. Next it will be demonstrated how LDA deals with this problem.

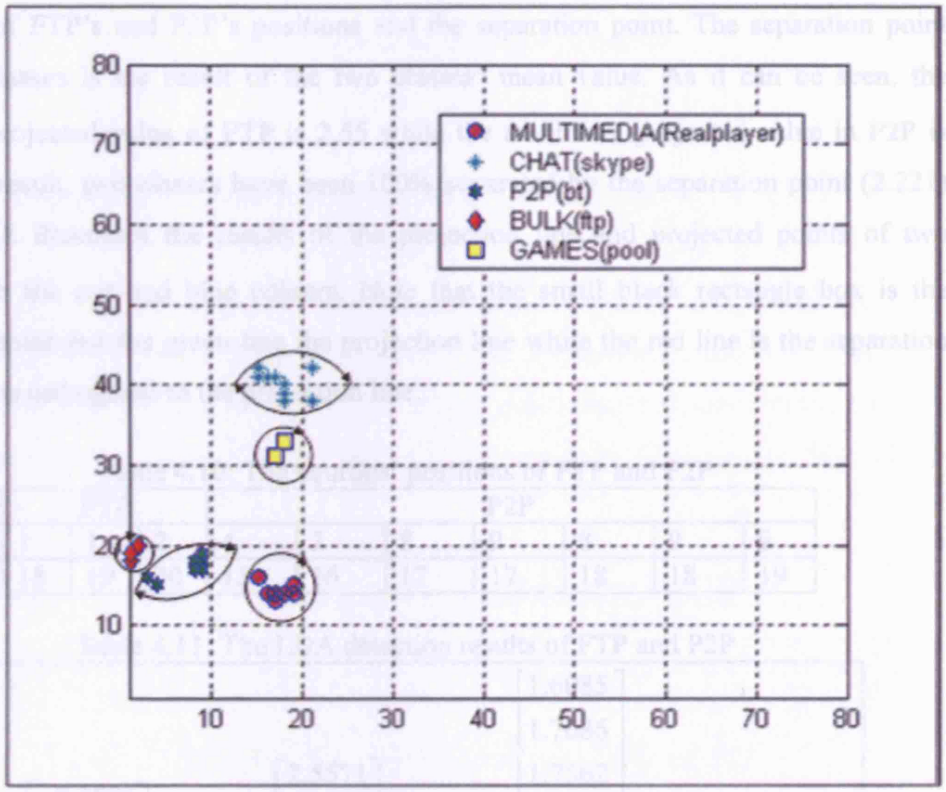


Figure 4.7: The classification results of five traffic categories

4.6.2 Classification results with Linear Discriminant Analysis

In Fig 4.7, five patterns have been produced and there are in total $C_5^2 = \frac{5!}{2!3!} = 10$ possible overlaps that may occur. Also, it shows that there are only three potential overlaps that need to be considered: Skype and Pool, FTP and P2P as well as P2P and Realplayer, for the pattern distance among other combinations are wide enough.

Partial Overlap case: LDA results between BULK (ftp) and P2P (bt)

Table 4.10 lists the positions of the responding neurons in the produced KNN map. Table 4.11 lists the outcomes of LDA calculation to the two traffic classes, in which v is the result of the transformation matrix, the other three vectors are the result of linear

projection of FTP's and P2P's positions and the separation point. The separation point $\bar{y}$ of two classes is the result of the two classes' mean value. As it can be seen, the minimum projected value of FTP is 2.55 while the maximum projected value in P2P is 2.06. As a result, two classes have been 100% separated by the separation point (2.221) of $\bar{y}$. Fig.4.8 illustrates the results of the projection line and projected points of two classes with the red and blue colours. Note that the small black rectangle box is the separation point and the green line the projection line while the red line is the separation line, which is orthogonal to the projection line.

Table 4.10: The neurons' positions of FTP and P2P

axis	FTP			P2P						
x	1	1	2	4	3	8	9	8	9	9
y	18	19	20	15	16	17	17	18	18	19

Table 4.11: The LDA detection results of FTP and P2P

$$v = \begin{bmatrix} -0.1001 \\ 0.1476 \end{bmatrix}; \quad y_{ftp} = \begin{bmatrix} 2.5571 \\ 2.7047 \\ 2.7522 \end{bmatrix}; \quad y_{P2P} = \begin{bmatrix} 1.6085 \\ 1.7086 \\ 1.7562 \\ 1.8139 \\ 1.8563 \\ 1.9038 \\ 2.0616 \end{bmatrix}; \quad \bar{y} = 2.221027;$$

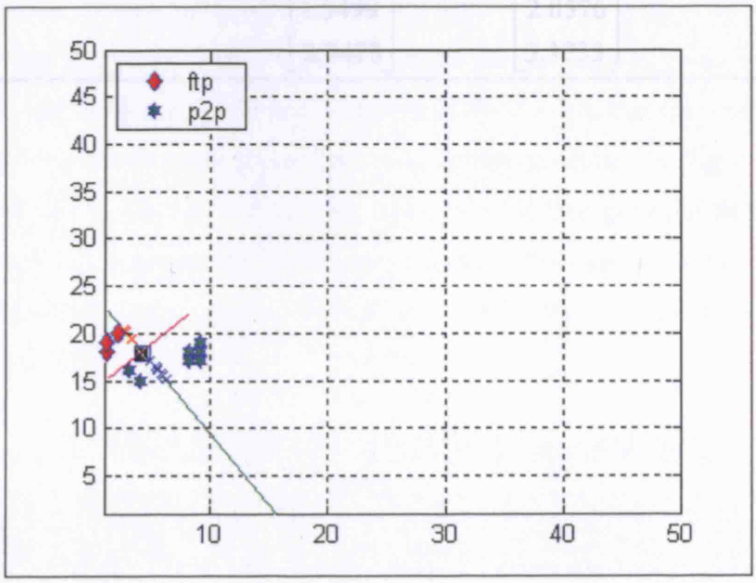


Figure 4.8: The LDA detection results of FTP and P2P

Partial Overlap case: LDA results between Realplayer and P2P

Similarly, Table 4.12 and 4.13 show two classes' neuron positions in the KNN map and the LDA projection results respectively. The LDA calculation results show that the projected vectors of Realplayer and P2P have been completely separated by the point $\bar{y}$ and hence no overlaps appear between two traffic classes along the projection line, which can be seen in the Fig.4.9.

Table 4.12: The neurons' positions of Realplayer and P2P

axis	P2P							Realplayer						
x	4	3	8	9	8	9	9	17	16	17	18	19	19	15
y	15	16	17	17	18	18	19	13	14	14	14	14	15	16

Table 4.13 The LDA detection results of Realplayer and P2P

$$v = \begin{bmatrix} -0.0841 \\ 0.2069 \end{bmatrix}; \quad y_{REALPLAYER} = \begin{bmatrix} 1.2589 \\ 1.2975 \\ 1.3816 \\ 1.4658 \\ 1.5043 \\ 1.5499 \\ 2.0478 \end{bmatrix}; \quad y_{P2P} = \begin{bmatrix} 2.7596 \\ 2.7666 \\ 2.8437 \\ 2.9665 \\ 2.0506 \\ 2.0576 \\ 3.1733 \end{bmatrix}; \quad \bar{y} = 2.163782;$$

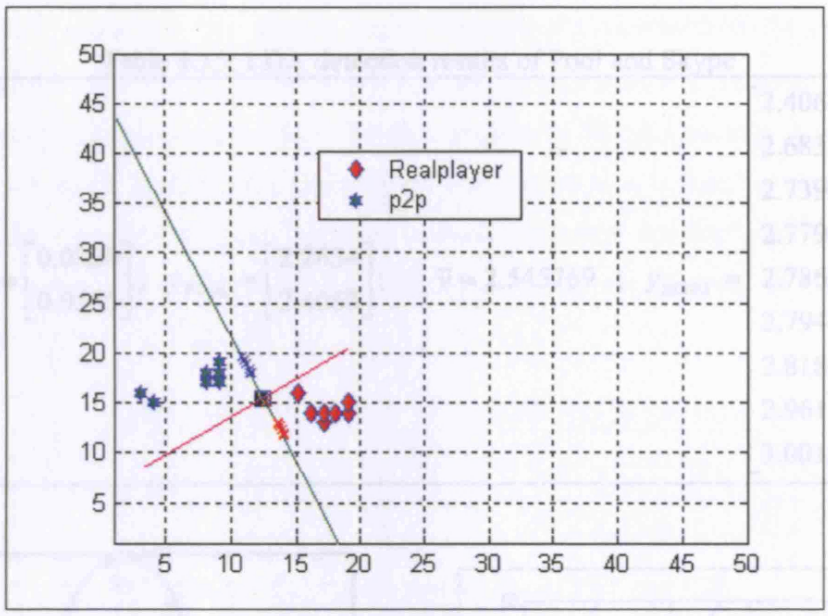


Figure 4.9: The LDA detection results of Realplayer and P2P

Complete Overlap Case: LDA results between CHAT (Skype) and GAMES (Pool)

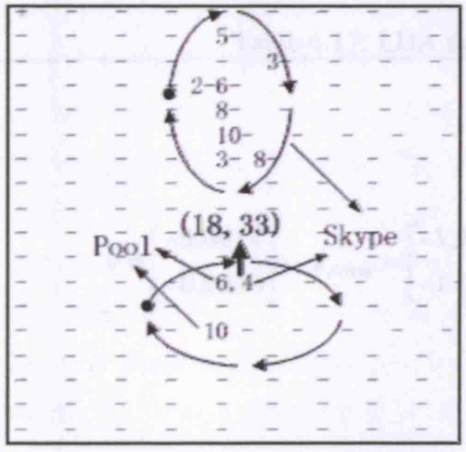
The LDA outcomes of Skype and Pool are different from the previous two cases as one neuron (18,33) has responded to both traffic classes, which causes the complete overlap issue. As shown in Table 4.15, the Pool's projected points have been entirely separated by $\overline{y}$. But, one (the point 2.406) of the Skype's projected values crosses another side of $\overline{y}$, which is supposed to be the Pool's area. One of the sample tests is given in Fig 4.10 (a). There are 49 group of Skype's data and 16 group of Pool's data that have been tested. The responding neurons can be categorised into two circles as shown in Fig 4.10 (a): one is Skype's area while the other is Pool's area. But, there are four group of Skype's data and six group of Pool's data appearing in the same neuron at the position of (18,33). Thus, the failure (misclassified) rate equals 4/49=8.16% while 91.86% of flows have been successfully classified by LDA.

Table 4.14: The neurons' positions of Skype and Pool

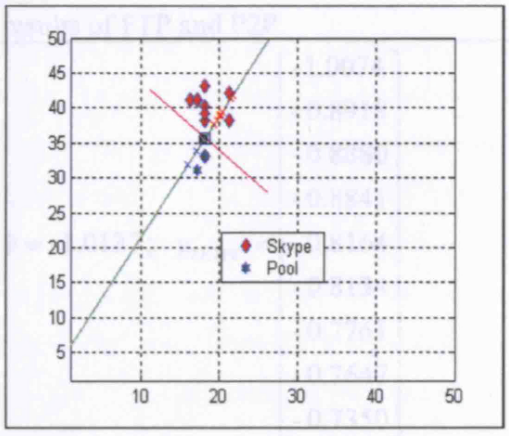
axis	Skype										Pool	
x	18	18	21	18	18	15	16	17	15	21	17	18
y	33	38	38	39	40	41	41	41	42	42	31	33

Table 4.15: LDA detection results of Pool and Skype

$$v = \begin{bmatrix} 0.0320 \\ 0.0555 \end{bmatrix}; \quad y_{POOL} = \begin{bmatrix} 2.2634 \\ 2.4063 \end{bmatrix}; \quad \bar{y} = 2.545769 \; ; \quad y_{SKYPE} = \begin{bmatrix} 2.4063 \\ 2.6837 \\ 2.7392 \\ 2.7796 \\ 2.7862 \\ 2.7947 \\ 2.8182 \\ 2.9611 \\ 3.0015 \end{bmatrix}$$



(a) A sample test of Skype and Pool



(b) The LDA detection results

(Note that there are 49 groups of Skype's data and 16 groups of Pool's data in (a))

Figure 4.10: The testing results of Skype and P2P

The second KNN maps with inputs of R1, R3 and R5

As discussed previously, the complete overlap issue is caused by the nature of the similarities in input data. Here, the inputs of R1, R3 and R5 are chosen to construct another map and the corresponding result is shown in Fig 4.11. Although the Skype's pattern is more scattered than the previous results, Table 4.16 shows that the complete overlap problem no longer exists. In addition to that, the outcomes of LDA testing results

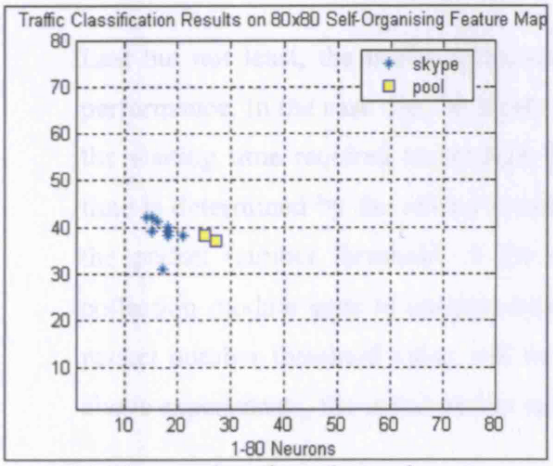
of Table 4.17 show that two classes have been entirely separated by the point $\bar{y}$. The reason why the selection of R1, R3 and R5 provides better results is because both R3 and R5 are time-related parameters. And, the time pattern of Skype's packets is paced by the human voice while that of Pool certainly is not related to a human's voice. Note that although the complete overlap has been solved, for other applications, the produced patterns of using {R1,R3,R5} will be more scatter than that of using {R1,R2,R4} like Realplayer as shown in Fig 3.25.

Table 4.16: The neurons' positions of Skype and Pool

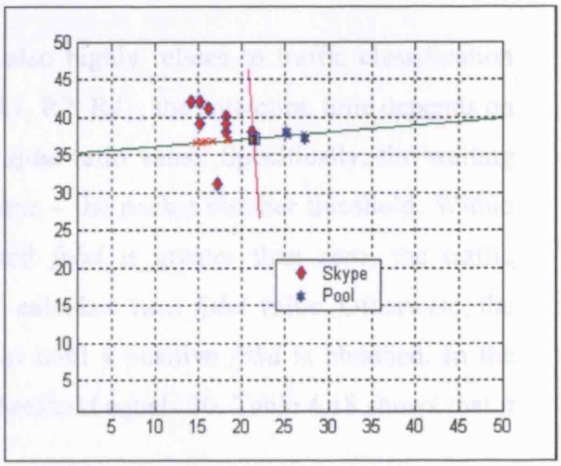
axis	Skype									Pool	
x	17	18	21	15	18	18	16	14	15	27	25
y	31	38	38	39	39	40	41	42	42	37	38

Table 4.17: LDA detection results of FTP and P2P

$$v = \begin{bmatrix} -0.0411 \\ -0.0038 \end{bmatrix};\ y_{POOL} = \begin{bmatrix} -1.2502 \\ -1.1718 \end{bmatrix};\ \bar{y} = -1.0137;\ y_{SKYPE} = \begin{bmatrix} -1.0074 \\ -0.8918 \\ -0.8880 \\ -0.8841 \\ -0.8164 \\ -0.8134 \\ -0.7761 \\ -0.7647 \\ -0.7350 \end{bmatrix}$$



(a) The results of an alternative map;



(b) The LDA detection results

Figure 4.11: Using the alternative map

4.6.3 The classification time

Apart from the detection accuracy, classification time is another important performance metric for a traffic classification scheme. For the proposed classification scheme, the classification time mainly includes the time spent on the KNN offline training and online learning, PCA and LDA analysing. The KNN offline training time varies in the selected KNN structures. For instance, to train an 80x80 KNN for 5000 iterations (reducing the learning rate to the predefined level), it could take from 2 minutes to over 10 minutes, depending on the performance of the chosen machine. It should be emphasised here that the offline training only requires to be performed once. After that, the relevant weight or structure size data will be stored in database for the online usage. The online KNN classification time is very fast from a few microseconds to hundreds of microseconds, varying for the different KNN structures and input numbers. It has been reported that the calculation of eigenvalue and eigenvector is time-consuming [JAM85] in PCA. But, this issue does not exist in this thesis, for PCA is only used once for the offline feature selection purpose. The PCA calculation time typically takes about a few seconds in Matlab. LDA is a lightweight based calculation for it is only used to distinguish the difference between two classes. Its calculation time is at microsecond level.

Last but not least, the traffic collection time also highly relates to traffic classification performance. In the case that the input set is {R1, R2, R4}, the collection time depends on the waiting time required to produce R1, the *fpbd* ratio value. Specifically, the waiting time is determined by the related threshold value – the packet number threshold. Within the packet number threshold, if the calculated *fpbd* is greater than zero, the traffic collection module goes to another iteration to calculate next *fpbd* value. Otherwise, the packet number threshold value will be doubled until a positive *fpbd* is obtained. In the above experiments, the initial packet number threshold equals 20. Table 4.18 shows that it

takes 179,473 microseconds[18] to calculate a Skype's *fpbd* value within every twenty packets. Clearly, the *fpbd* calculation time is varied in different type of flows. The more intensive a flow is, the less time an *fpbd* calculation takes. For example, compared with Skype, Table 4.19 shows the less time (120,263 microseconds) required to calculate a Realplayer's *fpbd* value. For online classification, a number of input sets of {R1,R2,R4} are usually needed as to produce robust results. Hence, the online classification time mainly depends on how many input sets will be tested. In the experiments presented in this thesis, the number was ranging from 10 to 25. This means that the classification time for Skype and Realplayer are: from 1.8 to 4.5 seconds and from 1.2 to 3 seconds respectively. In the case where the detected traffic belongs to long-live flow category (which means that the lifetime of traffic is greater than 5 seconds), the proposed classifier can be regarded as an early detection approach.

Table 4.18: The online detection data of Skype with packet number threshold =20

(Note, the timestamp is produced according to Unix's gettimeofday( ) function. It is expressed in elapsed seconds and microseconds since 00:00 Universal Coordinated Time, January 1, 1970.)

second	microsecond	SIP	DIP	Protocol	SPORT	DPORT	packet size
1187919426	6585	222. 122. 28. 195	144. 82. 193. 97	UDP	12340	37557	52
1187919426	22109	144. 82. 193. 97	222. 122. 28. 195	UDP	37557	12340	57
1187919426	26519	222. 122. 28. 195	144. 82. 193. 97	UDP	12340	37557	29
1187919426	36750	144. 82. 193. 97	222. 122. 28. 195	UDP	37557	12340	57
1187919426	47888	222. 122. 28. 195	144. 82. 193. 97	UDP	12340	37557	57
1187919426	62211	144. 82. 193. 97	222. 122. 28. 195	UDP	37557	12340	57
1187919426	66591	222. 122. 28. 195	144. 82. 193. 97	UDP	12340	37557	61
1187919426	86552	144. 82. 193. 97	222. 122. 28. 195	UDP	37557	12340	57
1187919426	86570	222. 122. 28. 195	144. 82. 193. 97	UDP	12340	37557	57
1187919426	93674	144. 82. 193. 97	222. 122. 28. 195	UDP	37557	12340	48
1187919426	107892	222. 122. 28. 195	144. 82. 193. 97	UDP	12340	37557	57
1187919426	120725	144. 82. 193. 97	222. 122. 28. 195	UDP	37557	12340	57
1187919426	126655	222. 122. 28. 195	144. 82. 193. 97	UDP	12340	37557	57
1187919426	131510	144. 82. 193. 97	222. 122. 28. 195	UDP	37557	12340	57
1187919426	146624	222. 122. 28. 195	144. 82. 193. 97	UDP	12340	37557	48
1187919426	156766	144. 82. 193. 97	222. 122. 28. 195	UDP	37557	12340	57
1187919426	166632	222. 122. 28. 195	144. 82. 193. 97	UDP	12340	37557	57

[18] It is equal to: 186058-6585.

1187919426	186058	144. 82. 193. 97	222. 122. 28. 195	UDP	37557	12340	61
1187919426	187154	222. 122. 28. 195	144. 82. 193. 97	UDP	12340	37557	57
1187919426	186058	144. 82. 193. 97	222. 122. 28. 195	UDP	37557	12340	57

Table 4.19: The online detection data of Realplayer with packet number threshold =20

Second	Microsecond	SIP	DIP	Protocol	SPORT	DPORT	Packet size
1187918708	710006	212. 58. 227. 104	144. 82. 193. 97	UDP	11302	6970	445
1187918708	710036	212. 58. 227. 104	144. 82. 193. 97	UDP	11302	6970	622
1187918708	718041	212. 58. 227. 104	144. 82. 193. 97	UDP	11302	6970	1045
1187918708	748544	212. 58. 227. 104	144. 82. 193. 97	UDP	11302	6970	276
1187918708	748667	212. 58. 227. 104	144. 82. 193. 97	UDP	11302	6970	390
1187918708	748696	212. 58. 227. 104	144. 82. 193. 97	UDP	11302	6970	211
1187918708	748951	212. 58. 227. 104	144. 82. 193. 97	UDP	11302	6970	442
1187918708	763446	212. 58. 227. 104	144. 82. 193. 97	UDP	11302	6970	152
1187918708	763641	212. 58. 227. 104	144. 82. 193. 97	UDP	11302	6970	413
1187918708	764734	212. 58. 227. 104	144. 82. 193. 97	UDP	11302	6970	663
1187918708	780243	212. 58. 227. 104	144. 82. 193. 97	UDP	11302	6970	916
1187918708	784515	212. 58. 227. 104	144. 82. 193. 97	UDP	11302	6970	241
1187918708	785051	212. 58. 227. 104	144. 82. 193. 97	UDP	11302	6970	410
1187918708	794549	212. 58. 227. 104	144. 82. 193. 97	UDP	11302	6970	280
1187918708	795206	212. 58. 227. 104	144. 82. 193. 97	UDP	11302	6970	412
1187918708	807767	212. 58. 227. 104	144. 82. 193. 97	UDP	11302	6970	352
1187918708	813427	144. 82. 193. 97	212. 58. 227. 104	UDP	6970	11302	109
1187918708	815251	212. 58. 227. 104	144. 82. 193. 97	UDP	11302	6970	396
1187918708	830229	212. 58. 227. 104	144. 82. 193. 97	UDP	11302	6970	789
1187918708	830269	212. 58. 227. 104	144. 82. 193. 97	UDP	11302	6970	994

As discussed above, the speed of the proposed classifier is another promising feature. The time spent on KNN training and PCA is only required once during the offline period while the online classification time needed by KNN and LDA is at the microsecond level. The proposed classifier is potentially an early classification solution for it often can give meaningful results before the flows are ended.

4.7 Chapter summary

This chapter explores the use of PCA and LDA to achieve highly accurate results in classification. It has been shown that PCA is an effective technique in the feature selection, for it is able to quickly identify which inputs contribute the least to the principal components in terms of the associated viabilities. Experimental results have demonstrated that the tight patterns have been produced according to the input selection using PCA. On the other hand, it has been shown that LDA plays an important role in dealing with the overlap problems between patterns. However, it is not sufficient when different applications coincide on the same neuron where LDA effectively tells how to draw the difference among different patterns with varying shapes and therefore makes the reading of different patterns possible. For the selected five categories, based on more than 100 measurements, the classification accuracy has already reached as high as 91.86% with the use of PCA and LDA. The reason for 8% misclassification is that whereas the inputs produced by PCA may be optimal as a whole, they may not be optimal for an individual application case. This problem can be solved by constructing a second KNN map. Take Skype and Pool for instance, the second map led to the classification results achieving 100% accuracy.

The speed of the proposed classifier is another promising feature. The time spent on KNN training and PCA is only required once during the offline period while the online classification time needed by KNN and LDA is at the microsecond level. The proposed classifier is potentially an early classification solution for it often can give meaningful results before the flows are ended.

Chapter 5

The End-to-End Bandwidth Estimation in IP Networks

5.1 Introduction

Bandwidth estimation in IP networks is important in many areas such as network monitoring, QoS-based routing, the development of time-sensitive applications, P2P networks etc. The traditional means for bandwidth estimation is through passively monitoring network nodes. In the IP networks, the cost of this is high, for it requires not only the cooperation among nodes within a large scale network but also introduces considerable overhead. Passive monitoring invokes a number of technical issues such as real-time estimation, the difficulty in deployment and policy constraints between different network operators etc. Recent years have seen a great many studies directed towards the active probing approach to bandwidth estimation. This approach, often lightweight and fast, only involves two end nodes along a network path, and it can be easily deployed and produces comparable results to passive monitoring.

This chapter reviews and discusses key aspects of active bandwidth estimation. It starts with an introduction to the basic definitions in the end-to-end bandwidth estimation, and is followed by a discussion how the bottleneck spacing effect, packet dispersion is used for the active network capacity measurement. Then, it goes on to discuss different forms of packet dispersions due to the impact of cross traffic in a practical networks, and how packet dispersions can be used for available bandwidth (avail-bw) estimation and introduces two models behind existing avail-bw estimation tools: the Probe Gap Model

and the Probe Rate Model. Next, the major avail-bw estimation tools will be discussed and compared.

5.2 Understanding of Available Bandwidth Concepts

5.2.1 Related Concepts

- **Bandwidth**

In the context of data networks, bandwidth often refers to the data rate that a network link can deliver [PDM03] and is typically measured in bits per second (bps). If a network link is able to deliver the maximum data D bytes within S seconds, then the bandwidth B of this link equals: $B = \frac{D \times 8}{S}$ *bps*. Bandwidth is often used to describe the data rate required by a network application or service. For instance, a voice application needs 8kbps bandwidth from networks.

- **The capacity and available bandwidth (*avail-bw*)**

There are two bandwidth metrics associated with a network path: the *capacity* C and the *avail-bw* A. The former is the maximum rate that a network path can deliver and the latter means the unused network capacity of a network path. For a given network path, the capacity C stays unchanged while the *avail-bw* A is changing from time to time. The minimum capacity link in a network path is called the *narrow link* while a link that has the least unused capacity is called the *tight link*. It should be noted that a *narrow link* may not be necessary the same as its *tight link* [DRM01][PDM03]. Another term that has been widely motioned is *bottleneck*, which could refer to both links (*tight link* and *narrow link*). In the case that the *narrow link* is the same as the *tight link*, it is also called that there is one single bottleneck on the path [SKK03][DRM01].

Let H be the total number of hops on a path, C_i is the capacity at link i, then this path's capacity C equals:

$$C = \min_{i=0 \ldots H} C_i \tag{5.1}$$

If $u_i^{\tau}(t_0)$ means the average utilization of link i during $(t_0, t_0+\tau)$, the *avail-bw* A of link i during $(t_0, t_0+\tau)$ can be expressed as:

$$A_i^{\tau}(t_0) = C_i(1-u_i^{\tau}(t_0)) \tag{5.2}$$

Consequently, the *avail-bw* A of a network path is [SBW05]:

$$A^{\tau}(t_0) = \min_{i=0 \ldots H} A_i^{\tau}(t_0) \tag{5.3}$$

It should be noted that the value of *avail-bw* is varied if it is measured in different layers. For example, the packet length at layer 2 is usually larger than that at layer 3[19], for packets at layer 2 have 38 bytes more (18 bytes for Ethernet header, 8 bytes for the frame preamble and other 12 bytes for the interframe gap). The majority of existing *avail-bw* tools are measuring *avail-bw* at the 2nd layer [PDM03] unless special instructions are given in advance.

5.3 The Rationale on Active Bandwidth Estimation Techniques

5.3.1 The Bottleneck Spacing Effect

The rationale for active network capacity or *avail-bw* estimation starts with the bottleneck spacing effect concept, which was originally stated in [JAC88]. Basically, as shown in Fig 5.1, a packet's transmission time will reach the maximum P_b at the bottleneck link and remain unchanged until the packet arrives at the sink, in which $P_r = P_b$. In addition, the

[19] The Maximum Transmission Unit (MTU) is 1500 bytes on Ethernet at layer 3.

time space P_b of a packet is proportional to its packet size but inversely to the capacity of the bottleneck link [LB01]. Thus, by sending a series of packets with different sizes, the capacity of a path can be derived through the analysis of the changes in P_b.

Figure 5.1: Packet-pair model (source from [JAC88])

(P_b - Transmission time at bottleneck, P_r - transmission time at receiver. The spacing between acks (returning packets) $A_s = A_b = A_r = P_b$. [JAC88])

5.3.2 The packet-pair technique and packet dispersion

The above section describes the bottleneck spacing effect in the case of using one probing packet. Such a spacing effect becomes even clearer under the packet-pair scenario. What the packet-pair technique does is to send two identical packets closely enough, as close as they can be queued at the bottleneck link so that two packets arrive at the sink with a time gap. Ideally, the time gap is equal to the bottleneck space as described above. However, such a time gap is usually referred to as packet dispersion, originally described in [JAC88]. The packet dispersion literally means the space between the last bits of the first packet and the second packet at the receiver [DRM01]. Related theoretical studies and evaluation of the packet pair technique can be seen in [KES91] [DRM01].

Based on the packet-pair technique, the bandwidth of the bottleneck can be calculated as follows [LB00][LB01].

Figure 5.2: Packet dispersion (source from [LB01])

To begin with, let t_l^k represent the transmission time of the packet k on link l, s^k be the packet size of the packet k and b_{l_b} be the bottleneck's bandwidth. As shown in Fig 5.2, two back-to-back packets start with the time space $(t_0^1 - t_0^0)$ and the space is $\frac{s^1}{b_{lb}}$ at the bottleneck and $(t_n^1 - t_n^0)$ at the sink. The packet dispersion $(t_n^1 - t_n^0)$ can be computed as follows:

$$t_n^1 - t_n^0 = \max(t_0^1 - t_0^0, \frac{s^1}{b_{lb}}) \tag{5.4}$$

Once the packet dispersion is known, the bottleneck bandwidth can be calculated below:

$$b_{bottleneck} = \frac{s^1}{t_n^1 - t_n^0} \tag{5.5}$$

It should be noted the above calculation is based on the assumption that the network node follows the FCFS (First Come First Served) queueing policy. Under the FCFS policy, it has been reported that cross traffic can seriously affect packet dispersions [LB01][MBG02b].

5.3.3 The diversities and evolution of packet dispersion along a network path

Due to the presence of cross traffic, the packet dispersion can appear in diverse forms. In [LB01], Lai et al summarised the four possible outcomes to packet dispersion (Fig 5.3).

Figure 5.3: Four Cases: A – D (source: [LB01])

Case A: It happens in ideal network conditions where there is no cross traffic interfering with probing packets and, therefore $s_{disperion} = s_{bottleneck}$.

Case B: There is cross traffic after the 1st probe packet but not before that, which leads to $s_{disperion} > s_{bottleneck}$.

Case C: There is cross traffic before the 1st probe packet but not after that, which leads to $s_{disperion} < s_{bottleneck}$.

Case D: This happens when there is a relatively large initial gap between two packets. In this case, probing packets will not be queued at the bottleneck link. It likely occurs in the high speed network environment.

It should be noted that the study of [LB01] missed another case, case E as discussed below.

- ***Case E and the related packet dispersion evolution***

Case E: In reality, it is more often that cross-traffic appears not only before the 1st packet but after the 1st packet too. In such circumstances, the changes of packet dispersions are rather complex as discussed below.

Using the same notations from [DRM04], let d_i^1 and d_i^2 be the queueing delay of packet one and two caused by cross traffic and τ_i be the probing packet's transmission time at link *i*, then the dispersion Δ_i at link *i* falls into the following two circumstances [DRM04]:

1) Fig 5.4 illustrates the packet dispersion changes between two probing packets from hop i-1 to hop i. Note that packet 1 is placed on the left as the x-axis stands for time. As the two probing packets have the same size, so their transmission delays (τ_{i-1}) are equal. Let Δ_{i-1} be the packet dispersion at hop i-1. The bottom of Fig 5.4 shows the delay and transmission time of two packets at hop i. Note that the delay d_i^1 starts exactly at the moment when packet 1 leaves hop i-1 while entering hop i. Hence, if $d_i^1 + \tau_i$ is higher than dispersion Δ_{i-1}, then the new packet dispersion Δ_i equals $\tau_i + d_i^2$. Under this circumstance, the changes of packet dispersions depend on both the transmission time τ_i and the delay time d_i^2.

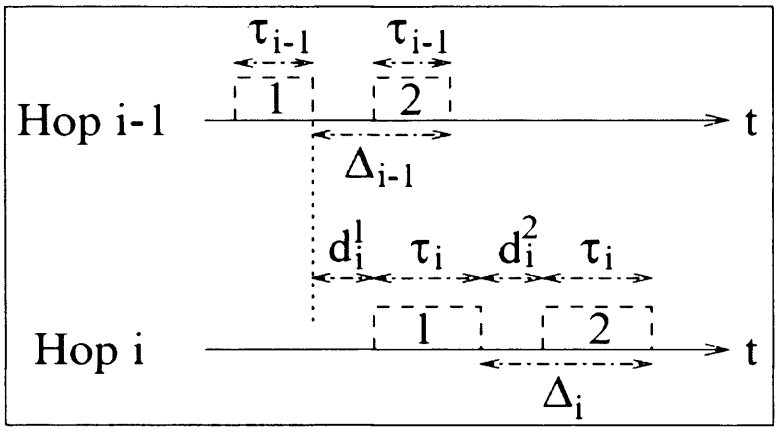


Figure 5.4: The first case when $d_i^1 + \tau_i > \Delta_{i-1}$

2) In the second circumstance, if $d_i^1 + \tau_i$ is lower than the packet dispersion Δ_{i-1}, namely, $\tau_i + d_i^1 < \Delta_{i-1}$, then the dispersion $\Delta_i = \Delta_{i-1} + (d_i^2 - d_i^1)$ (Fig.5.5).

Proof: Let x be the shared part between Δ_{i-1} *and* Δ_i, *we can get* $x = (\Delta_{i-1} - d_i^1 - \tau_i)$. *In addition to that, as x also equals* $(\Delta_i - d_i^2 - \tau_i)$, *the dispersion at hop i is:* $\Delta_{i-1} + d_i^2 - d_i^1$

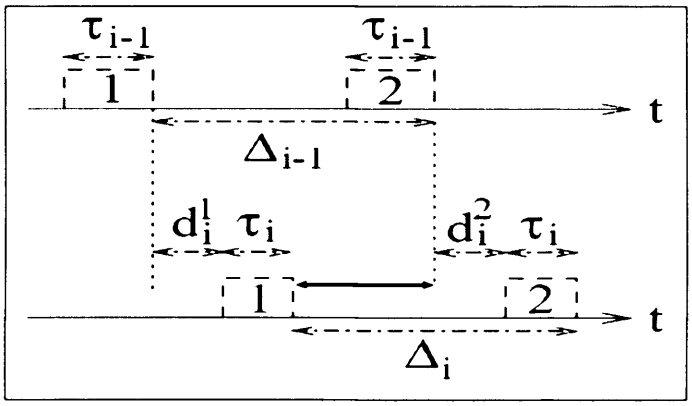


Figure 5.5: The second case when $\tau_i + d_i^1 < \Delta_{i-1}$

5.3.4 Using packet dispersion for bandwidth estimation

As discussed above, packet dispersion appears in various forms that can be categorised into five different cases. Of these cases, the case E is the most likely to happen. It is clear that the elimination of such deviations is an important issue for network capacity estimation. One of the earliest studies to address this issue was carried out by Carter et al [CC96a]. They investigated the impact of cross-traffic on packet dispersion by sending variable packet sizes. The conclusion was that correct estimates would correlate with each other, however, incorrect estimates would exhibit less or a lack of correlation [CC96a][CC96b]. Accordingly, a filter has been developed to discard the incorrect estimates through the *union* and *intersection* methods. The *union* technique is to combine the overlapping estimates into a histogram using *set union* while the *intersection* is to calculate the overlaps of estimates [CC96b]. However, it is difficult to choose an appropriate bin width (called the avail-bw estimate resolution) in a histogram [LB00][LB01]. As an extension and improvement to Carter et al's work, the Kernel Density Estimation[20] based filter has been proposed by Lai et al in [LB00][LB01].

$$\int_{-\infty}^{\infty} K(t)dt = 1 \;\Leftrightarrow d(x) = \frac{1}{n}\sum_{i=1}^{n} K\left(\frac{x - x_i}{cx}\right) \;\Leftrightarrow K(t) = \begin{cases} 1+t & ;t \le 0 \\ 1-t & ;t > 0 \end{cases} \qquad (5.6)$$

For example, as shown in Fig 5.6, among the three cases (A, B, C from Fig 5.6), the highest density appears on the case A.

[20] This (KDE) has been also used in section 3.5.2 (see Appendix for related implementations).

Figure 5.6: Clustering feature of dispersion (source:[LB01])

When it comes to *avail-bw* estimation, the way of dealing with diversity of packet dispersions is rather different from that used in capacity estimation. Instead of eliminating the changes of packet dispersion that is caused by cross-traffic, such changes are used to infer the related cross-traffic rate. Since the capacity of a network path can be known beforehand using tools such as *Pathrate* [DRM04], *Bprobe* [CC96b] and *Nettimer* [LB01] etc, if the cross-traffic rate[21] is known then related *avail-bw* can be easily obtained. Thus, the question is how to devise such mechanisms that compute avail-bw estimation through the observation of changes in packet dispersion. This is discussed in the following sections.

[21] Note that the term of the cross-traffic rate has the same meaning of competing-traffic rate.

5.4 Towards Available bandwidth Estimation

5.4.1 Can avail-bw be estimated according to the packet dispersion of packet trains?

Cprobe, the first tool using the packet dispersion for avail-bw estimation, is developed according to the assumption that the dispersions of a packet train are inversely proportional to the *avail-bw, as stated in* [CC96b][JD02]:

> *"By bouncing a short stream of echo packets*[22] *off of the target server and recording the time between the receipt of the first packet and the receipt of the last packet, we can measure the presence of competing traffic on the bottleneck link. Dividing the number of bytes sent by this time yields a measure of available bandwidth".*

Another tool called *Pipechar* was developed with the same assumption [JD02]. However, this assumption has been proven wrong. In [DRM01][DRM04], Dovrolis et al study the relationship between the distribution of the estimated bandwidth *B(N)* [23] and corresponding packet-train dispersions. It was found that the distribution of *B(N)* was multimodal when N is small (Fig 5.7-a), but it gradually becomes unimodal when N is large enough. Such a unimodal distribution will stay unchanged even if N increases further. In addition, the center of the unimodal distribution is independent of N (Fig 5.7-b) and is referred to as the *asymptotic dispersion rate (ADR)* in [DRM01]. Given a network path, the ADR is a static metric. Hence, the dispersion of packet trains does not measure actual *avail-bw* but ADR.

[22] They are produced by ICMP protocols.
[23] Each train with N back-to-back packets

Figure 5.7: The multimodal and ADR unimodal distribution ([DRM01])

5.4.2 Probe Gap Model and Probe Rate Model

Although tools like *Cprobe* and *Pipechar* have not actually measured avail-bw but ADR as discussed above, the time gaps between probing packets do relate to *avail-bw* [JD02][SKK03], since the time gaps are affected by competing traffic. Recent studies [HS03][SKK03] show that avail-bw measurement can be realised by analysing time gaps rather than packet dispersions. First of all, there is a need to distinguish the difference between the concepts of packet dispersion and packet gap. Packet dispersion literally means the spacing effect under the condition that the initial gap is required to be as small as possible and stays unchanged during a measurement period. Packet gap refers to the time space between two successive packets at the receiver, but the initial gap is not required to be as small as possible and will be changed after every run. In this sense, the packet dispersion can be regarded as a special case of the packet gap concept. But the measurement of time gaps provides information required by avail-bw estimations. An avail-bw tool uses such information to infer the competing traffic and the turning point, which will be discussed below.

The recent avail-bw estimation tools can be categorised into two classes according to the main approaches underlying the estimation techniques [SKK03][CMC05]. The first class is called the Probe Gap Model (PGM), which is realised by exploiting the packet gap information to calculate the competing traffic rate. *Spruce* [SKK03], *IGI (Initial Gap Increasing)* [HS03]and *Delphi* [RCR00] are tools developed under the PGM. The second class is the Probe Rate Model (PRM) that utilizes trains of packets sent at different probing rates. Example of the tools to realise PRM includes such as *TOPP (Train of Packet Pair)* [MBG00], *Pathload* [JD02], *BART (Bandwidth Available in Real-Time)* [EKE06] etc.

5.4.2.1 Probe Gap Model (PGM)

The PGM is based on two assumptions. One is the single bottleneck assumption on the estimated path. The other is that the queue between successive packets is not empty. Since changes to the output packet gap is caused by cross-traffic, the PGM assumes that the time to transmit the cross traffic is $g_O - g_I$ when $g_O > g_I$ [SKK03]. Given a path's bottleneck capacity is C, the competing traffic rate is $\frac{g_O - g_I}{g_I} \times C$. Consequently, *avail-bw* A can be calculated by subtracting the cross traffic rate from capacity and is calculated as follows:

$$A = C \times \left(1 - \frac{g_O - g_I}{g_I}\right) \tag{5.7}$$

This equation explains why the PGM does work in a single bottleneck and does not work if there are multiple bottlenecks *(which equally mean multiple Cs)* in the probed path. Apparently, the cross-traffic rate can not be inferred from Equation (5.7) while there is no queue in existence between a pair of probing packets.

The turning point concept

A network link can be regarded as a multiplexed channel consisting of two slots. One is occupied by cross-traffic while the other is empty, namely the avail-bw slot. Given the assumption that the probed link follows the fluid model, an avail-bw slot will stay rather stable during a short period of measurement time. If the initial gap is so small that the avail-bw slot is overflowed, the queue will be built up in the tight link and the delay will be introduced for probing packets, which leads to the output gap being larger than the initial gap. However, we are able to set the initial packet gap with a value so that probing traffic is exactly filled into the *avail-bw*'s slot and it does not overflow, theoretically, cross traffic will not interfere with probing packets. At such a measurement point, the output gap g_O is equivalent to the initial gap g_I. In [HS03], the study shows that after the turning point, if the initial gap is increased further, the result of $g_O = g_I$ still holds. According to Equation (5.7), the result of the cross-traffic rate will be zero after the turning point and correspondingly the measurement should be terminated. This approach has been used in *IGI* [HS03] and related results show that the bandwidth measured by *IGI* is the competing traffic before the turning point.

5.4.2.2 Probe Rate Model (PRM)

As discussed above, the turning point is a point when the avail-bw slot of a path is fully occupied but not overflowed, which equally means that the probing traffic is the same as avail-bw when a turning point occurs. The PRM is operated under such a principle. With the PRM, the probing traffic is sent out with different rates and the avail-bw estimation is realised by searching for the turning point. There are three different approaches to realise the PRM in terms of the methods for the turning point detection:

1) The first one is simply realised by the comparison of the output gap and the initial gap. When $g_O = g_I$, the turning point is found. *PTR* (*packet transmission rate*) is an example of this approach.

2) The second one is based on the fact that when the sending rate of probing traffic is higher than *avail-bw*, queues will be built up in the tight link, which delay corresponding probing traffic. But, if the sending rate is lower than *avail-bw*, the arrival rate is equal to the sending rate. Therefore, the turning point is realised by the comparison between the arrival rate and sending rate. With this approach, the probing packets typically are sent with the initial gap in a decreasing manner. Tools like *TOPP* and *BART* are examples using this approach.

3) Another approach is realised according to the changes of one way delays (*OWD*) of probing packets. As Jain et al stated that when the probing rate becomes higher than *avail-bw*, queues appear and the OWD starts to increase as well [JD03][JD02]. Thus, the turning point actually is the point when the one way delays starts to grow. In this approach, the sending rate is typically sent in the decreasing manner. The tools *Pathload* and *wget* [AAP06] are realised with this approach.

Comparisons between PGM and PRM

Note that the underlying measurement principle of PGM and PRM is the same in the way that they are both realised by analysing time gaps and measuring turning points. But, the way of calculating avail-bw is different: PGM is based on competing traffic rate while PGM is calculated according to probing rate. The PGM only works in the single bottleneck path and it will fail in a multi-hop network path environment, for multiple bottlenecks exist [RRB03][SKK03][HS03]. This issue does not apply to PRM. But, PGM is usually less intrusive than PRM [SKK03].

5.5 The Major Avail-bw Estimation Tools Review

This section discusses the major avail-bw tools and gives relevant analysis for each tool. The commercial tools are not included here as they are not open to access and details have not been published.

5.5.1 *Avail-bw* tools with the PGM

- ***IGI, Spruce and Delphi***

IGI is a tool based on the single-hop gap model [HS03]. In this model, the output gap consists of two time segments. One is the time to process the probing packet while the other is the time to process the competing traffic, which arrives between the two probing packets (within g_I). The former equals $\frac{s}{B_C}$ while the latter is equivalent to $\frac{R_C \times g_I}{B_C}$, where B_C is the bottleneck capacity, s is the probing packet size and R_C is the cross traffic rate. Hence, the output gap equals:

$$g_O = \frac{s}{B_C} + \frac{R_C \times g_I}{B_C} \tag{5.8}$$

As the output gap g_O can be measured at the end host, the competing traffic rate can be inferred according to Equation (5.7). To deal with the dynamics and the burstiness of the competing traffic, *IGI* sends a number of packet trains for the g_O calculation instead of a single packet pair. In [HS03], the impact of packet size and the length of packet trains on *IGI* also have been investigated and drew the following conclusions. The best range of packet sizes for *IGI* is between 500 and 700 bytes. *IGI* underestimates avail-bw if using smaller packet sizes, and overestimates *avail-bw* with larger packet sizes. Regarding the length of a packet train, the results show that shorter packet trains cause a wider range of *avail-bw* estimates and needs more probing phases (probing iterations) in order to converge the best initial gap value for the turning point. The validation of *IGI/PTR* was carried out over a number of network paths between University campus networks and commercial networks [HS03]. It has been reported that *IGI/PTR* is a very fast tool usually taking 2-3 seconds and yields the comparable accuracy in comparison with *Pathload.*

Spruce is a lightweight avail-bw estimation tool, for the inter-pair gap is set as the exponentially distributed gap, which is much larger than the time gap between a pair of packets. As a result, *Spruce*'s measurement is operated with a Poisson sampling process [SKK03]. *Spruce* was evaluated in a real network environment and its validation was through comparison with the Multi-Router Traffic Grapher (MRTG) data. The designer of *Spruce* stated that *Spruce* was more accurate than *Pathload* and *IGI*. In addition, *Spruce*'s overhead is about 300 KB per measurement, which is low overhead in comparison to that of *Pathload* (typically 5MB). Strauss et al argue that one of the major reasons for *Spruce*'s improvement lies in that *Pathload* and *IGI* may disturb concurrent TCP flows because packet trains from *Pathload* and *IGI* do not the follow the Poisson process [SKK03].

Delphi appeared earlier than both *IGI* and *Spruce*. Unique to Delphi is the use of exponentially spaced packet trains and the MWM (*multifractal wavelet model)* to infer the cross-traffic [RCR00]. It is evaluated under *ns2,* and experimental results conclude that Delphi produces accurate estimates at high link utilisation levels while at low utilization, it overestimates the cross traffic. Note that *Spruce* sets the inter-pair gap in an exponential increasing manner, which is different from Delphi's setting.

5.5.2 *Avail-bw* tools with the PRM

- ***Pathload, TOPP and BART***

Pathload is based on the idea called *Self-Loading Periodic Streams* (SLoPS), which belongs to the self-induced approach [JD02]. In *Pathload*, the turning point detection is realised by monitoring the variations of one way delays of packet streams. The source sends a number (K=100) of equal-sized packets to the receiver at a certain rate R. When R is larger than avail-bw, the queue will be built up and subsequently the one way delays of the probing packets will keep growing. Otherwise, the one way delays (OWD) will not increase. By adjusting the sending rate and monitoring the changes of OWDs, *Pathload*

tries to make the stream rate R close to avail-bw, which is realised by an iterative algorithm similar to binary search [PDM03] as discussed below.

Pathload first sends a fleet of N streams, each stream consisting of K equalled-size packets[24]. For each N-unit stream, it partitions the K-OWD measurements into $\Gamma = \sqrt{K}$ groups. After that, *Pathload* starts to calculate the mean $\hat{D}_i$ of OWDs in each group and two statistical metrics called *Pairwise* Comparison Test (*PCT*) and *Pairwise* Difference Test (*PDT*) for the trend detection.

$$A_{PCT} = \frac{\sum_{k=2}^{\Gamma} I(\hat{D}^k > \hat{D}^{k-1})}{\Gamma - 1} \tag{5.9}$$

$$A_{PDT} = \frac{\hat{D}^{\Gamma} - \hat{D}^{k-1}}{\sum_{k=2}^{\Gamma} | \hat{D}^k - \hat{D}^{k-1} |} \tag{5.10}$$

where I(x) is a boolean function defined as:

$$I(x) = \begin{cases} 1 & (\hat{D}^k > \hat{D}^{k-1}) \\ 0 & otherwise \end{cases} \tag{5.11}$$

As stated in [JD02], the 'increasing trend' is indicated when PCT value >0.66 while the 'non-increasing trend' is reported when PCT <0.54. Otherwise, the OWDs are in an 'ambiguous trend'. The PDT value reflects the net 'increasing trend'. It returns a value between -1 and 1, where -1 indicates a strongly decreasing OWD trend while 1 indicates a strongly increasing OWD trend. The next step for *Pathload* is to iteratively compare the estimated R with two values[25]: R^{min} and R^{max}. R^{min} refers to the highest rate that has been shown to be less than the available bandwidth up to a certain point, while R^{max} is the lowest rate that has been shown to be higher than the available bandwidth up to that point. Finally, estimates will be converged according to a user-specified resolution ω and, the

[24] The length of a fleet equals : NxK packets.
[25] Note that R here refers to estimated bandwidth range used to infer desired avail-bw.

target *avail-bw* is found when $R^{max} - R^{min} \leq \omega$. Experimental results show that *Pathload's* estimates are very close to MRTG's avail-bw reading with only about 0.5 Mbps difference [JD02].

TOPP is realised by comparing the sending rate with the arriving rate. It is able to detect multiple congestible links. A link is called congestible when the receiving rate is larger than the link surplus rate. It assumes that the probing traffic and cross traffic each receive their proportional share of the link capacity C under the FCFS policy. Let u stand for the probing traffic rate, X be the cross-traffic rate and C be the link capacity, then the receiving rate r is [MBG00]:

$$r = \frac{u}{u + X} C \tag{5.12}$$

The ratio of the probing rate and the receiving rate indicates if there is congestion happening during the measurement period. If the ratio equals 1, there is no congestion. Otherwise, the ratio is a first-order polynomial in u:

$$\frac{u}{r} = \begin{cases} 1 & (u \leq C - X) \\ \frac{X}{C} + \frac{1}{C} u & (u > C - X) \end{cases} \tag{5.13}$$

Thus, *avail-bw* can be evaluated by capturing the turning point where the ratio starts to deviate from unity by varying the probing rate u. So as not to flood the network, TOPP increases the sending rate by ΔR after each run until it meets the turning point. By plotting the probing rate against the arriving (output) rate r, the capacity C and the avail-bw A can be inferred by the use of the linear regression technique [MBG02]. TOPP solves the hidden hop issue as it is able produce all the surplus bandwidth of congestible links. But, one major problem in TOPP is that it is based on the Smallest Surplus First (SFF) assumption in a multiple congestible link path, otherwise its estimates are incorrect [MBG00]. More discussions on the hidden hop and the surplus problems are given in Appendix D.

BART is one of the latest tools which appeared in 2006 [EKE06]. It is realised by the measurement of the packet strain, a concept derived from the delay variation of a packet pair. The delay variation of a pair of packets means the difference between the arriving time gap (g_I) and leaving time gap (g_O) at one hop, and a packet strain (ε) is defined as:

$$\varepsilon_i = \frac{g_O - g_I}{g_I} \tag{5.14}$$

Provided the packet size equals *s*, then the equation 5.13 can be rewritten as:

$$\frac{u}{r} = \frac{s/g_I}{s/g_O} = \frac{g_O}{g_I} = 1 + \varepsilon_i \tag{5.15}$$

where *u* stands for probing traffic rate while *r* is receiving rate. This equation suggests that there is no congestion on the path when ε=0 while the probing rate is higher than *avail-bw*, the packet strain ε is a positive value [EKE06]. On the other hand, by combining the equation 5.13 with 5.15 together, the packet strain also can be expressed as:

$$\varepsilon = (\frac{x}{c} - 1) + \frac{1}{c}u = \alpha + \beta * u \tag{5.16}$$

where $\alpha = x/c - 1$ and $\beta = 1/c$ [MBG00]. Thus, the target avail-bw A is given by:

$$A = -\frac{\alpha}{\beta} \tag{5.17}$$

Fig 5.8 plots the packet strain against the probing traffic rate and shows that the packet strain starts to increase from the available bandwidth point $(-\frac{\alpha}{\beta}, 0)$.

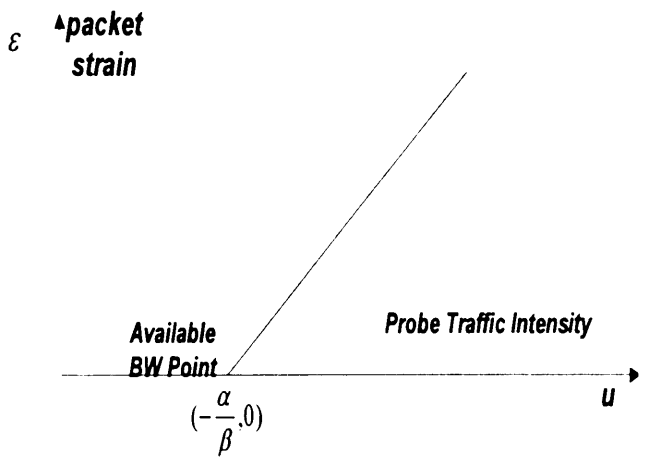


Figure 5.8: The packet strain against the probing traffic intensity

In order to yield varying packet strains, BART uses a number of packet trains. The Kalman filter has been used in BART as a special 'twist' to infer the parameters α and β for the *avail-bw* calculation.

- **Discussions on the avail-bw tools**

Among the PGM-based tools, the major difference lies in the varying settings of the initial gap between the intra-pair or inter-pair packets such as the linearly decreased intra-pair gap in *IGI*, the exponentially distributed inter-pair gap in *Spruce* and the exponentially distributed intra-pair gap in Delphi. The varying settings lead to different models and techniques for cross-traffic rate measurement. However, the PGM based tools face the single-bottleneck issue as discussed in 5.4.2.2 and incorrect estimation results may occur in a multiple-hop path.

The differences between the PRM based tools arise from two sources. One is the range of metrics employed for the turning point detection, such as the one way delay parameter used in *Pathload*, the ratio of the probing rate and receiving rate used in TOPP and the packet strain used in BART etc. The other is the range of algorithms or techniques that have been used for avail-bw estimation. For example, *Pathload* uses the binary search

algorithm, TOPP uses the linear regression method while BART uses the Kalman filter. Both TOPP and BART confront the SFF (Smallest Surplus First)[26] issue in a path that includes multiple congestible points [JD02][MBG02b][EKE06].

The validation of an avail-bw tool include can be realised in three different ways: simulation [JPW07], the comparison with other tools or the comparison with MRTG reading [JD02]. The simulation based validation is probably the least reliable, but simulations provide the most flexibility in the testbed topology. Validation through comparison with other major tools or with MRTG reading over the real network paths is the most common approach. Up to now, the validation results from current tools and the chosen testbed show much variation. For example, *IGI/PTR* is said to be superior to *Pathload* [HS03] while *Spruce* is said to outperform both *IGI* and *Pathload* [SKK03]. However, Shriram et al [SMH05] present a comprehensive study that evaluates a number of publicly available tools on a high-speed testbed and real networks[27] with OC-48 and GigE paths. Shriram et al concluded that *Pathload* was the most accurate tool [SMH05]; this was further investigated by Antoniades et al [AAP06].

[26] See Appendix.

[27] The Abilene Network: the end machines had a 1Gbps connection while the rest of links in the path had either 2.5 or 10 Gbps capacities.

5.6 Chapter summary

In summary, this chapter has presented a comprehensive analysis of underlying rationales, techniques and models in the active bandwidth estimation field as shown in Fig 5.9.

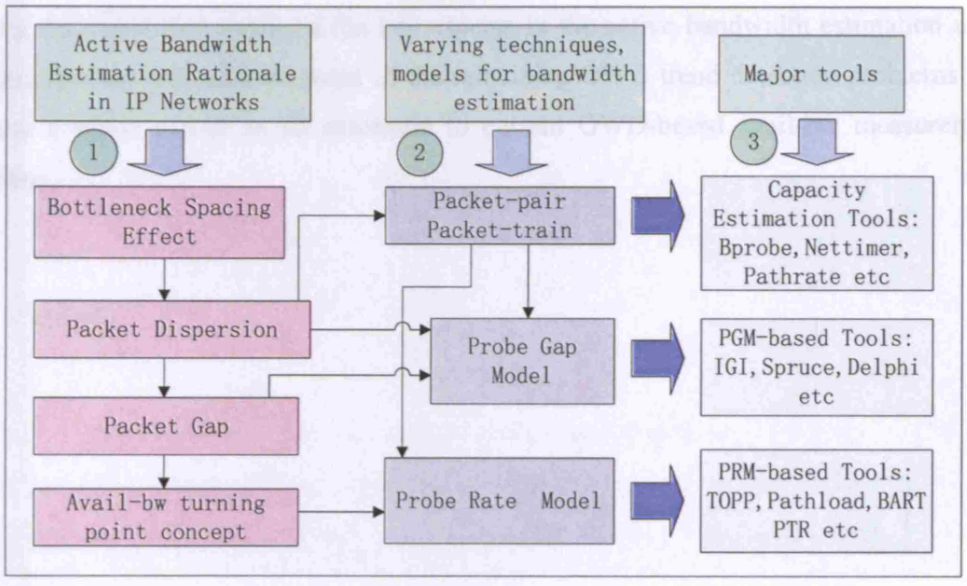


Figure 5.9: An overview of Chapter 5

First of all, the rationale of the active bandwidth estimation started with the idea called bottleneck spacing effect proposed by Jacobson. Then the packet dispersion concept appeared and subsequently a number of studies on the use of the packet-pair technique for network capacity estimation emerged. Tools such as *Bprobe*, *Nettime* and *Pathrate* etc are examples based on the packet-pair or packet-train techniques. With the similar approach used in *Bprobe*, Cater et al developed *Cprobe* for avail-bw estimation by measuring packet dispersions of packet trains. However, as *Dovolis* et al point out that what *Cprobe* measures actually is another metric called ADR rather than *avail-bw*. Existing solutions for active avail-bw estimation are categorised into two classes: Packet Gap Model and Probe Rate Model. As discussed in the aforementioned section, by using different

intra-pair packet gaps or inter-pair packet gaps, a variety of PGM-based tools appeared lately but the single-bottleneck issue still remains unsolved. The PRM is a self-induced approach. Of these PRM based tools, *Pathload* has been regarded as the most accurate tool, in part for it is based on the OWD measurement, which avoids the hidden hop and SFF (Smallest Surplus First) issues.

Having discussed and analysed the key aspects in the active bandwidth estimation area, the next chapter will address some of the remaining OWD trend detection problems and propose a novel model as an extension to current OWD-based avail-bw measurement solutions.

Chapter 6

Pathpair: A fast Avail-bw Measurement tool with the Asymptotic OWD Comparison Model

6.1 Introduction

The preceding chapter investigated related concepts and techniques in the active avail-bw estimation area during the past decade and addressed the key aspects of the Probe Rate Model and the Probe Gap Model as well as corresponding techniques used in current major avail-bw estimation tools.

This chapter is going to explore a new model - the Asymptotic OWD Comparison (AOC) model for avail-bw estimation. The major goal of this model is to enhance the OWD trend detection reliability and robustness. This model essentially is an extension and enhancement to those avail-bw estimation methods that are realised by the detection of the OWD variations i.e. *Pathload* [JD02] and *Wget* [AAP06]. An AOC based avail-bw estimation tool - *Pathpair* is proposed. The name *Pathpair* means that the core algorithms to detect the OWD trend are realised by a pair of comparisons. The comparisons are not only between successive OWDs but also between actual OWDs and the asymptotic OWDs. The following sections are structured as follows. Section 6.2 discusses the motivations behind this work and presents the objectives of *Pathpair*. Section 6.3 introduces the basic

concepts and definitions of the AOC model and presents the new algorithm and three statistical metrics for the OWD trend detection; in addition the location of the avail-bw turning point is discussed. Section 6.4 addresses the design issues in *Pathpair*. Section 6.5 gives the overall implementation architecture of *Pathpair*. Section 6.6 presents the detailed experimental outcomes as well as the validation results.

6.2 Aims and Motivations

6.2.1 Aims and Objectives

The primary goal of this chapter is to develop a fast available bandwidth estimation tool based on the OWD trend detection. The slow measurement speed hampers the applicability of existing avail-bw estimation tools. For example, as stated in [EBET03], existing bandwidth estimation tools are often not reliable and precise enough for short-time estimates to assist congestion control or the improvement of TCP slow-start. Some major tools' measurement times are listed below according to [HS03][SMH05]:

- *Pathload*: 7.2 to 22.3 seconds
- Patchchirp: 5.4 seconds
- Iperf: 10.0 to 10.2 seconds
- *Spruce*: 10.9 to 11.2 seconds
- *IGI/PTR*: 2 to 3 seconds

Thus, this chapter aims to explore an avail-bw estimation scheme to improve the estimation speed while also preserving comparable results for existing major tools like *Pathload*:

- To design a new measurement model that is able to deal with dynamics of OWDs and yield the reliable ATP detection results.

- To design a new algorithm that can quickly detect an avail-bw turning point.
- To prototype the AOC model and develop an avail-bw estimation tool (*Pathpair)* based on the real network environment.
- To validate *Pathpair* through comparing its detection results with existing major tools.

6.2.2 Remaining Problems

Existing solutions for the OWD trend detection and avail-bw turning point identification is through the comparison between consecutive OWDs within a very short period like the PCT and PDT based algorithm [JD02], typically within a few hundreds of microseconds[28], which is so short that the outcomes of an ATP (avail-bw turning point) detection are often incorrect due to the dynamics of OWDs as discussed below.

The dynamics of OWDs is mainly caused by three factors: the burst cross-traffic, the effect of interrupt coalescence and the dynamically changing performance at the probing machines. First, the short burst cross-traffic often results in a very short queue which leads to some OWDs in the increasing trend, which does not reflect the overall trend of OWDs. Recently, as discussed in Chapter 5, a number of studies have shown that cross traffic have a serious impact on packet dispersion or packet gap [CC96b][DRM01] [LB00]. Such an impact applies equally to corresponding OWDs. This can be illustrated as follows:

Let $s(i)$ and $s(i-1)$ be the sending time of packet i, i-1 at the sender respectively;

Let $r(i)$ and $r(i-1)$ be the receiving time of packet i, i-1 at the receiver respectively;

[28] It depends on the sending traffic rate and is subject to the probed path.

Let g_I and g_O represent the input gap and the output gap between a pair of packets;

Let $owd(i)$ and $owd(i-1)$ be the OWD of packet i, i-1;

Let *offset* refer to the time difference between the sender and receiver;

Then, it follows that $g_I = s(i) - s(i-1)$, $r(i) = s(i) + owd(i) + offset$ and the output gap g_O is equal to:

$$\begin{aligned} g_O &= r(i) - r(i-1) \\ &= s(i) + owd(i) + offset - s(i-1) - owd(i-1) - offset \\ &= s(i) + owd(i) - s(i-1) - owd(i-1) \\ &= s(i) + owd(i) - (s(i) - g_I) - owd(i-1) \\ &= owd(i) - owd(i-1) + g_I \end{aligned}$$

Therefore,

$$owd(i) = g_O + owd(i-1) - g_I \qquad (6.1)$$

Hence, according to the equation 6.1, it is clear that cross traffic and packet dispersion or packet gap have comparable affects on OWDs.

The second factor is interrupt coalescence[29] (IC) and it has been reported in [CMC05][PD04][JD02]. Basically, interrupt coalescence is designed to reduce the number of network interrupts from an NIC (Network Interface Card) within short time intervals. It is defined as [PD04]:

> *"IC is a technique in which NICs attempt to group multiple packets, sent or received in a short time interval, in a single interrupt."*

In the presence of IC, packets are buffered at the NIC and will not be processed until the interrupt time expires. Within one interrupt period, the first arrival packet experiences the

[29] The term IC is also called Context Switch (CS) in a number of studies [PD04][JD02].

longest queueing delay while the last packet experiences the least. As a result, in the presence of IC, OWDs are more dynamically changing and looks like Fig 6.1, which may lead to the incorrect trend detection result. In *Pathload*, once IC presence is detected, the decreasing parts of OWDs are simply discarded. This means significant part of measurement information is thrown away at the IC elimination stage [CMC05], for it is hard to tell which part of OWDs should be discarded. It should be noted that although the IC effect could seriously distort the OWD trend, the overall trend of OWDs is still rising[30].

Figure 6.1: OWDs in 100-packet train with and without IC (source: [PD04])

The third factor affecting the dynamics of OWDs arises from the changing performance of the probing machines. For instance, the CPU will slow down to load socket functions like *sendto() or recvfrom()* when the probing machines are busy [SKK03][CMC05]. This problem typically happens in a sender machine since the sender reads the system clock in a polling loop, due to the narrow time gaps between sending packets [SKK03]. Often this issue causes OWDs to fall as the sending rate is reduced[31]. In *Spruce*, once a processor that is being scheduled for other applications is detected, *Spruce* discards the current task and starts a new polling process [SKK03]. *Pathload* produces wrong OWD trend detection results when probing machines' performance is degrading (see section 6.3.2.2).

[30] In the case that the probing rate is larger than avail-bw.
[31] This is because the queueing will drop when the sending rate declines.

It should be noted that the OWD dynamics could also be caused by other factors such as the properties of physical links and routers, the number of hops along a network path and the protocols used for delivering traffic.

6.3 The Asymptotic OWD Comparison Model

6.3.1 Basic Concepts

This section readdresses the one way delay (OWD) concept, the OWD trend, the *asymptotic* OWD and avail-bw turning area under the asymptotic OWD comparison model.

6.3.1.1 One way delay (OWD), the OWD trend and *Asymptotic* OWD

Definition 1: ***One way delay*** refers to the transmission time of a probing packet between two nodes *SND* (sender) and *RCV (receiver)* along a network path.

Let t_{RCV} stand for the timestamp on a probing packet at the *RCV* while t_{SND} is the timestamp at *SND*. Then, the corresponding OWD equals $(t_{RCV} - t_{SND})$. The value of an OWD could be either positive or negative. If two end machines have been synchronised, an OWD's value will be positive. However, even if a relative OWD appears negative, the result of the OWD trend detection will not be affected, for only the relative values between OWDs are required for avail-bw estimation.

Definition 2: ***One way delay trend*** is the evolving trend of OWDs within a measurement period under the self-induced model.

If the probing traffic is not operated under the self-induced model [DRM01], the trend of OWDs could appear to be more random. Under the self-induced model, as discussed in Chapter 5, OWDs will not start to grow until an avail-bw turning point (ATP) occurs. In other words, the shape of an OWD line consists of two parts during a measurement period: a level line before an ATP and a straight rising line after an ATP. Hence, the ATP detection is equally the same as the finding of the conjunction point of two lines. In particular, due to the effect of cross-traffic, the impact of IC and the changing performance at both ends (i.e. the sendto() and *recvfrom()* functions) as mentioned above, the OWD trend fluctuates dynamically as shown in Fig 6.2. Therefore, if the OWD trend detection is realised by examining successive OWDs within a short time, the detection results may often be inaccurate. To deal with this issue, the asymptotic OWD concept is introduced.

Figure 6.2: OWD variations (Source:[JD02])

Definition 3: ***Asymptotic*** **One Way Delay** is not a real OWD of a probing packet but a value that is calculated by taking all the OWDs into account.

One simple solution to calculate a new AOWD is to use the mean between an OWD and its previous mean OWD value. For example, the i^{th} asymptotic OWD is equal to:

$$AOWD_N = \frac{\sum_{i=1}^{N} OWD_i}{N} \tag{6.2}$$

$$\text{where } i > 1 \text{ and } OWD_i = \begin{cases} 2 \times AOWD_{i-1} & (OWD_i - AOWD_{i-1}) > 2 \times AOWD_{i-1} \\ \quad or & (OWD_i - AOWD_{i-1}) < (-2 \times AOWD_{i-1}) \\ OWD_i & otherwise \end{cases}$$

To make sure that a new AOWD will not be dominated by any very high or very low *OWDs*, the value of a chosen OWD is limited to the range described in Equation (6.2). Fig 6.3 shows a schematic of this range. Note that the first value of AOT_1 is the same with OWD_1.

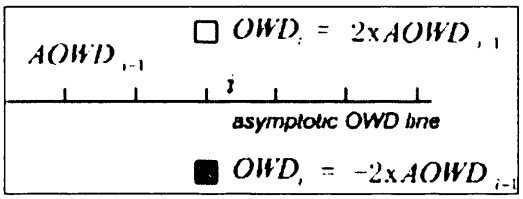


Figure 6.3: The calculation of an asymptotic OWD

Hence, the asymptotic OWDs (AOWD) are changing at a much slower pace than the actual OWDs. The longer a measurement tool is continued, the less impact the current OWD's will have on the *asymptotic OWD*.

6.3.1.2 Avail-bw turning point *and Avail-bw turning area*

The meaning of the avail-bw turning point concept varies in different measurement tools. For example, in *IGI/PTR*, an avail-bw turning point refers to the point when $g_I = g_O$. In this thesis, an avail-bw turning point is regarded as the conjunction point of two lines where the *asymptotic OWD line* simulates the first line. Then, the avail-bw turning point is defined as:

Definition 4: Avail-bw turning point refers to the point when OWDs start to grow from the *asymptotic OWD line.*

It is apparent that *asymptotic OWDs* will also start to ascend after an ATP but with the slower rising pace than actual OWDs, for *asymptotic OWDs* are averaged by previous *OWDs* (before ATP) as shown in Fig 6.4. As a result, after an ATP, there will be a notable difference between real OWDs and corresponding AOWDs. Consequently, through the comparison between real OWDs with AOWDs, the turning point can be detected.

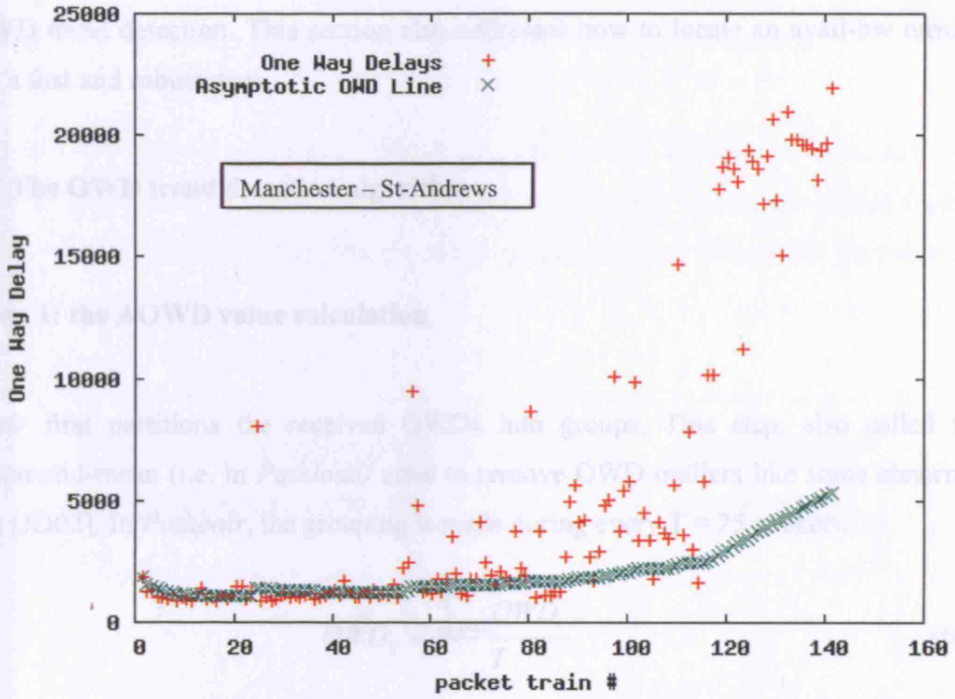


Figure 6.4: The asymptotic OWD line and actual OWDs

(Note that two different averages have been used to produce this figure, these are discussed in section 6.6.1)

However, in the practical network environment, such an absolute avail-bw turning point does not exist. Rather, as shown in Fig 6.2 and Fig 6.4, an ATP looks more like a small region. So the term - *avail-bw turning area (ATA)* is more appropriate than the concept of ATP. The term ATA is similar to the name of 'grey region' [DRM04]. It is clear that, within an avail-bw turning area, although some of OWDs may appear below the *asymptotic OWD line,* the majority of OWDs will appear above.

6.3.2 The Asymptotic OWD Comparison (AOC) Model

This section first discusses how to detect the OWD trend with the AOC algorithms. Then, it presents three different cases that illustrate how to use the proposed AOC algorithm for

the OWD trend detection. This section also addresses how to locate an avail-bw turning area in a fast and robust way.

6.3.2.1 The OWD trend detection algorithm

- **Step 1: the AOWD value calculation**

Pathpair first partitions the received OWDs into groups. This step, also called the 'partition-and-mean (i.e. in *Pathload)* aims to remove OWD outliers like some abnormal OWDs [JD02]. In *Pathpair*, the grouping is made during every T = 25 packets.

$$\hat{OWD}_T = \frac{\sum_{i=1}^{T} OWD_i}{T} \tag{6.3}$$

Note that in case of that the usleep() or select() functions have been used at the sender for releasing the CPU processor, the delay caused by these functions also needs to be taken into account at the receiver. Having computed $\hat{OWD}_T$, a new AOWD value can be easily obtained according to Equation (6.1).

- **Step 2: The comparison between $\hat{OWD}_T$ and AOWD**

As the difference between an $\hat{OWD}_T$ and AOWD may range widely, it will be transformed into a smaller range data (-1,1) to create comparison value φ:

$$\varphi = \frac{\hat{OWD}_T - AOWD}{\sqrt{\left(\hat{OWD}_T - AOWD\right)^2 + AOWD^2}} \tag{6.4}$$

For instance, in Fig 6.4, due to the dynamics of OWDs, some $\hat{OWD}_T$ may appear abnormally (i.e. having very large values at packet train #22), which will affect the OWD trend detection (typically affect the result of A_{SOT} as discussed below). But, after transforming the difference into the φ 's value, this matter is resolved.

- **Step 3: the OWD trend detection through three statistical metrics**

The trend detection is performed iteratively. The packet number in each iteration is defined by parameters K and T. The length of $K \times T$ usually equals one packet train. For example, if K=5 and T=20, the total packet length equals one 100-packet packet train as shown in Fig 6.5.

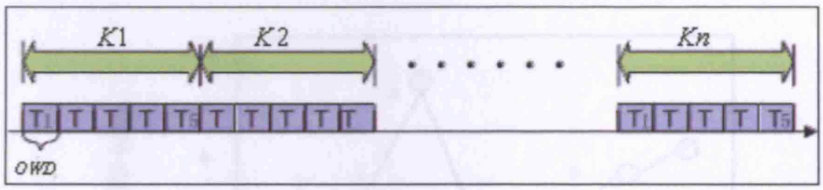


Figure 6.5: The parameters T and K

Given K group of $\hat{OWD}_T$ s, the OWD trend detection is realised by examining three statistical metrics. The first metrics is the Sum of Trend (SOT), calculated below:

$$A_{SOT} = \int_{x=\hat{owd}_1}^{\hat{owd}_K} f(x)dx \tag{6.5}$$

If the overall trend of OWDs is increasing, A_{SOT} will be a value that is larger than zero.

Note that if the majority of φs are negative, the value of an A_{SOT} could still be positive if a small number of φs have very high values (Fig 6.6). This may well lead to the false increasing trend outcome. To overcome this problem, the second metric – Positive Trend Checking (PTC) is introduced:

$$A_{PTC} = \frac{\sum_{i=1}^{K} I(\varphi_i >= 0.1)}{K} \tag{6.6}$$

where I(x) is one if x holds and zero otherwise. In *Pathpair*, OWDs are regarded as having a trend above the AOWD value only when an $\hat{OWD}_T$ is at least 10% larger than corresponding AOWD value. *Pathpair* judges whether OWDs are above an AOWD line or not according to A_{PTC} and A_{SOT}. In *Pathpair*, it was found empirically that the detected OWDs are above AOWD line when $A_{PTC} \geq 0.6$ and $A_{SOT} > 0$. Note that these settings are experiment based and it is clear that the high A_{PTC} imposes strict conditions and may affect the measurement speed.

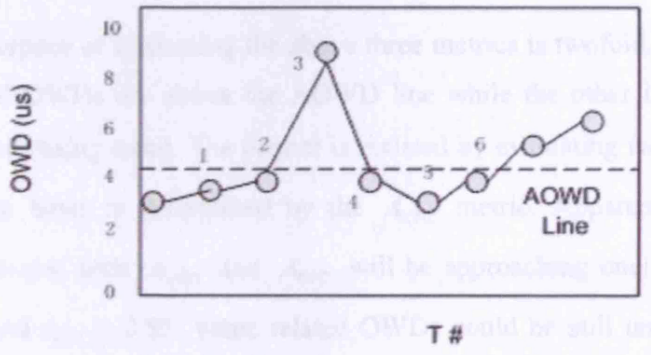
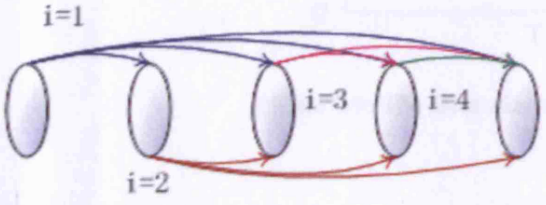


Figure 6.6: A_{SOT} calculation affected by a couple of large OWDs

The third metric is called Complete Trend Comparison (CTC), which is to examine the rising strength of the OWD trend. For K group of $\hat{OWD}_T$ s, there are $K(K-1)/2$ possible trend results in total (Fig.6.7). The CTC is realised by repeating the comparison between φ_i $(1 \leq i \leq K-1)$ and φ_j $(i+1 \leq j \leq K)$. $I(x)$ is one if x holds; otherwise it equals zero.

$$A_{CTC} = \frac{\sum_{i=1}^{K-1} \sum_{j=(i+1)}^{K} I(\varphi_j > \varphi_i)}{\frac{1}{2} \times K(K-1)} \tag{6.7}$$



$i=1; j \in [2,5]$
$i=2; j \in [3,5]$
$i=3; j \in [4,5]$
$i=4; j \in [5,5]$

Figure 6.7: A CTC process with $K=5$

If the result of A_{CTC} is less than 0.5, the detected OWDs are in a decreasing trend. Otherwise, the higher value of A_{CTC}, the stronger is the increasing trend of OWDs. In *Pathpair*, the rising trend is reported when $A_{CTC} \geq 0.55$.

In summary, the purpose of examining the above three metrics is twofold. One is to make sure that the actual OWDs are above the AOWD line while the other is to ensure that OWDs are in an increasing trend. The former is realised by evaluating two metrics A_{SOT} and A_{PTC} while the latter is determined by the A_{CTC} metric. Apparently, if a strong increasing trend occurs, both A_{PTC} and A_{CTC} will be approaching one. However, even when $A_{PTC} \geq 0.6$ and $A_{CTC} \geq 0.55$, some related OWDs could be still under the AOWD line, or say, $A_{SOT}<0$. For example, from the group 1 to group 5 of OWDs in Fig.6.8, although $A_{PTC} \geq 0.6$ and $A_{CTC} \geq 0.55$, the corresponding increasing trend is very weak since $A_{SOT}<0$. Therefore, in such a case, the 'non-increasing' trend is reported in *Pathpair*. In the next section, more discussions on how to use the proposed three metrics to detect the OWD trend are given.

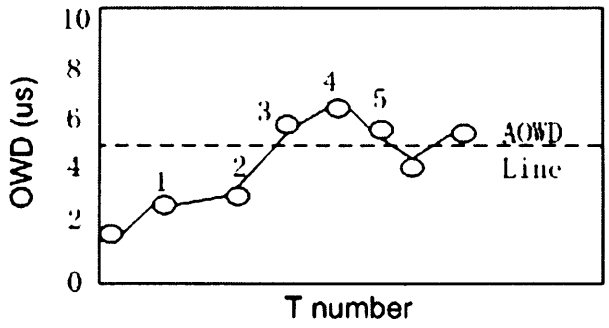


Figure 6.8: The illustration of why A_{PTC} is required.

6.3.2.2 Discussions on the improvement of the AOC algorithm

This section illustrates how to detect the OWD increasing trend with the AOC algorithm and shows related enhancement through a comparison with *Pathload*'s PCT and PDT algorithms.

- *Case 1: OWDs affected by interrupt* coalescence

The first case, as shown in Fig 6.9, presents the typical case when OWDs experience the presence of interrupt coalescence (IC) at the receiver when the probing rate is larger than avail-bw. In this case, the expected trend result should be 'increasing'. First, let us use *Pathload's* algorithm to calculate the PCT metric, which can be easily calculated to be 0.37 (4/9) as there are four rising trend between successive OWDs (Fig-6.3.8). Hence, a 'non-increasing' trend will be reported in *Pathload*. This explains in part why the decreasing part of OWD needs to be discarded in *Pathload* as discussed before [JD02] .

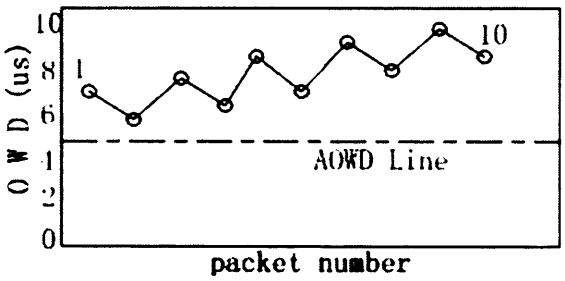


Figure 6.9: The IC case

In the case of AOC algorithms, the detection result is different with the PCT's metric. First of all, since all the OWDs are above the AOT line, $A_{PTC}=1$ and $A_{SOT}>0$. In addition, as there are 34 OWDs showing the increasing trend (Table 6.1) and according to equation 6.8, the A_{CTC} can be calculated from:

$$A_{CTC}=\frac{34}{10(10-1)/2}\approx 0.75;\quad (K=10);$$

Table 6.1: A_{CTC} results

i=1	Increasing no=6	i=2	Increasing no=8	i=3	Increasing no=5
i=4	Increasing no=6	i=5	Increasing no=2	i=6	Increasing no=4
i=7	Increasing no=1	i=8	Increasing no=2	i=9	Increasing no=0
Total	Increasing no=34				

Therefore, the detected trend outcome is 'increasing' with the AOC algorithm.

- *Case 2: OWDs affected by burst traffic*

The burstiness of cross-traffic has a serious impact on the OWD detection result. Basically, a short period of burst cross-traffic may result in a small queue, leading a few OWDs to very high values. Let us assume that the first high OWD is caused by burst cross-traffic in Fig 6.10.

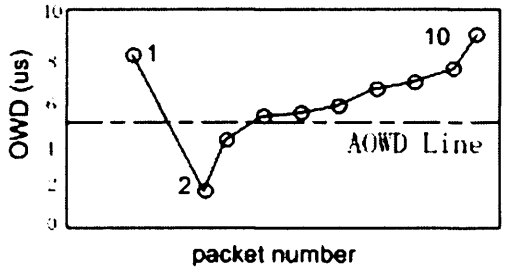
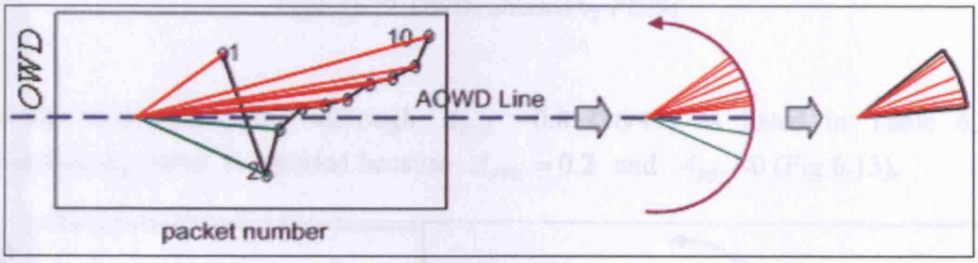


Figure 6.10: OWDs affected by cross-traffic

Since there are nine OWDs from 2 to 9 showing the rising trend, PCT=0.9. However, the difference between the 1st OWD and the 10th OWD is so small that PDT is less than the minimum value 0.45 required for an increasing trend to be reported [JD02]. Thus, the 'non-increasing trend' is reported in *Pathload*. Nevertheless, if *Pathload* starts the calculation with the 2nd OWD, PDT>0.56 and an 'increasing trend' is reported. As such, in *Pathload*, the trend detection result sometimes depends on which OWD is chosen as the 1st one for the PDT calculation. This problem does not exist in the AOC approach. Firstly, it is clear that $A_{SOT}>0$ (Fig 6.11). Secondly, $A_{PTC}=0.7$ as there are seven *OWD*s

out of ten OWDs, which are above the AOWD line. Furthermore, as there is only one decreasing trend between the 1st OWD and 2nd OWD, then A_{CTC} is calculated to be:

$$A_{CTC} = \frac{10(10-1)/2-1}{10(10-1)/2} \approx 98\%; \quad (K=10);$$



(This diagram illustrates the positive value of A_{SOT} as indicate by the sector.)

Figure 6.11: An illustration of the A_{SOT} calculation

- *Case 3: OWDs affected by the performance degradation at sender machine (PDSM)*

When experiencing performance degradation like CPU being busy, a sender machine usually is unable to load related socket functions like *sendto*() as fast as expected. In such circumstances, the sending probe speed may decrease to a lower rate than actual avail-bw while the related trend often seems to be increasing if only by comparing consecutive OWDs.

In Fig 6.12, let us assume that the probing rate is lower than avail-bw due to the effect of PDSM, most of the related OWDs are below the AOWD line (Fig.6.12). In *Pathload*, the 'increasing trend' will be reported as PCT=0.55 (5/9) and PDT >0.55.

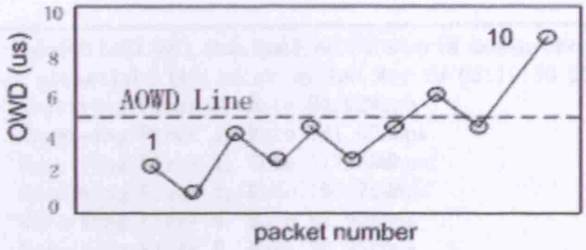
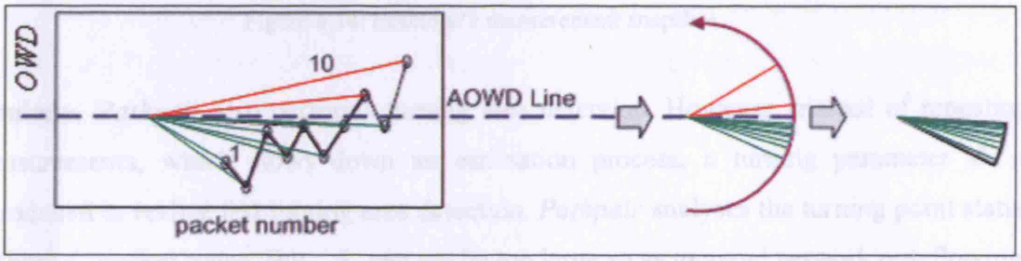


Figure 6.12: OWDs affected by *PDSM*

With the AOC algorithm, although A_{CTC} =0.8 (36/45) as listed in Table 6.2, a 'non-increasing trend' is reported because $A_{PTC} = 0.2$ and A_{SOT}<0 (Fig 6.13).



(This diagram illustrates the negative value of A_{SOT} as indicate by the sector.)

Figure 6.13: An illustration of the A_{SOT} calculation

Table 6.2: A_{CTC} result

i=1	increasing no=8	i=2	increasing no=8	i=3	increasing no=5
i=4	increasing no=5	i=5	increasing no=2	i=6	increasing no=4
i=7	increasing no=2	i=8	increasing no=1	i=9	increasing no=1
total	Increasing no= 36				

6.3.2.3 Locating avail-bw turning area and calculating avail-bw

As discussed in section 6.3.1, an avail-bw turning point appears more like a turning area. To locate an ideal turning area, one solution is to repeat the measurement a number of times like *Pathload*. For example, the *Pathload* measurement outcomes captured between Kent and Queen Mary University Planetlab nodes are shown in Fig 6.14. Such an iterative process is also called turning area detection

```
Receiver planetlab1.nrl.dcs.qmul.ac.uk starts measurements
at sender planetlab2.ukc.ac.uk on Tue May 29 02:19:50 2007
        Receiving Fleet 0, Rate 94.02Mbps
        Receiving Fleet 1, Rate 141.00Mbps
        Receiving Fleet 2, Rate 117.50Mbps
        Receiving Fleet 3, Rate 105.75Mbps
        Receiving Fleet 4, Rate 99.90Mbps
        Receiving Fleet 5, Rate 96.95Mbps
        Receiving Fleet 6, Rate 47.00Mbps
        Receiving Fleet 7, Rate 47.00Mbps
        Receiving Fleet 8, Rate 70.50Mbps
        Receiving Fleet 9, Rate 82.25Mbps
        Receiving Fleet 10, Rate 76.35Mbps
        Receiving Fleet 11, Rate 79.30Mbps
```

Figure 6.14: *Pathload's* measurement snapshot

Similarly, *Pathpair* also performs turning area detection. However, instead of repeating measurements, which slows down an estimation process, a turning parameter δ is introduced to realise fast turning area detection. *Pathpair* analyses the turning point status within δ packet trains. But, δ can not be too large so as to avoid network overflowing. During δ packet trains, if 60% of OWDs are showing the rising trend, then a turning area is located. Due to the reason that OWDs just starts to rise from the AOWD line during the first 30% period of this turning area, the beginning 30% of probing rate is used to approximate avail-bw in *Pathpair*. During this period $(0.3 \times \delta_T)$, if the probing packet number equals *N* and packet size is *S*, then the detected avail-bw is equal to:

$$A = \frac{N \times S}{0.3 \times \delta_T} \tag{6.8}$$

where δ_T is the total transmission time for $\delta \times K \times T$ packets.

- **The AOC Algorithm - pseudo codes**

The main pseudo code of the AOC algorithm is given below. There are four sub functions in this code:

calculate_mean_OWD is performed iteratively within every 25 packet as to deal with outliers of OWDs.

A_SOT_calculation is activated once K equals 5 and to calculate SOT values.

A_PTC_calculation, similarly, is activated once K equals 5 and to calculate PTC values.

A_CTC_calculation is used to calculate CTC values.

The array of *owd_inc_trend* is Boolean type and stores the trend detection results. The final aim of this code is to judge if an avail-bw turning area exist. If so, the avail-bw calculation function *Goto_availbw_calculation( )* is triggered. Otherwise, the code goes to the next iteration.

```
Pathpair_AOC_Algorithm
{
        /* Initialisation */
        #define GROUP_NUMBER_T 25;
        #define PERIOD_NUMBER_K  5;
        #define TURNING_AREA_NUMBER_δ  10;
        n=0;T=0;K=0;
        while(recvfrom(client)){
        n++;
        if((n%GROUP_NUMBER_T)==0)
          {
                calculate_mean_OWD();
                T++;
                if(T%PERIOD_NUMBER_K){
                        A_SOT_calculation();
                        A_PTC_calculation();
                        A_CTC_calculation();
                        if(A_SOT>0)&&(A_PTC>=0.6)&&(A_CTC>=0.55)
                        owd_inc_trend[T]=TRUE;
                        else owd_inc_trend[T]=FAULSE;
                        K++;
                        if(K>=TURNING_AREA_NUMBER_δ){
                        if(60% in the detected OWDs showing increasing trend)
                                Goto_availbw_calculation();
                    }
                }
          }
        }
}
```

6.4 *Pathpair's* selections on the probing mode, protocol and related parameters

When it comes to the design of an avail-bw tool, a number of issues need to be addressed:

- Which probing mode is more suitable for *Pathpair* between the single-end or double-end mode?
- Which protocol should be chosen between TCP and UDP?
- The selection of packet size and the probe train length.

6.4.1 The selection of single-end or double-end mode

The single-end mode (SEM) only requires the execution of measurement codes at one end of the probed path. Under the SEM, as the timestamp information at the receiver is unknown, the round-trip delay is used for the avail-bw calculation. There are various techniques to realise the SEM. The first technique to realise the SEM is to use the ICMP protocol. Basically, a probing packet is sent out with the preset TTL; this forces the destination to sends ICMP error packet (56 bytes [STE94]) back to the source. Tools like *Cprobe* and *Nettimer* are examples of using this technique. The second technique is based on the fact that TCP server will automatically reply with RST packets once the client sends TCP SYN packets to the server, like *Sprobe* [SGG02], *Sbing* [SAV99]. Another technique is to make use of a limited advertised window and to use the generation of paced 'fake' ACKs like *wget* [AAP06]:

> *"If the client acknowledges only one MSS with each ACK and it advertises a window of only one MSS, then the server will be forced to send one MSS upon received each ACK, as long as the server has at least MSS bytes available in the send socket buffer."*

The double-end mode (DEM) requires the measurement software to be deployed at both path ends. The majority of existing tools are currently realised with the DEM.

Clearly, the single-end model is more scalable and more easily deployed than the double-end model. However, if heavy traffic exists in the reverse path, the yielded estimates are likely to be inaccurate since related RTTs will be distorted. Moreover, the server may not respond in a timely way to send the packets back to a client. For example, many routers have the restriction of acknowledging ICMP echo messages to prevent the malevolent use of ICMP functionalities [LB00]. Obviously, the DEM is free from such kind of issues. Moreover, an ABWE tool can fully manage and control the packet size and

the probing rate under the DEM. Since the AOC algorithm, to locate the avail-bw turning area, requires a period of continuously stable probing rate, which the SEM can not provide, the DEM is chosen for the design of *Pathpair*.

6.4.2 The selection of TCP and UDP

TCP stems from Network Control Protocol (NCP) based on the *DARPA* (*Defence Advanced Research Project Agency*) research network. TCP is the connection-oriented protocol, which sets up the logic circuit channel before each flow transmission [RFC793] and implements the congestion control functionalities with varying algorithms like TCP-Tahoe and TCP-Reno [JAC88][JAC90a][JAC92] under different versions. UDP is a connectionless protocol and does not implement the congestion control functionalities. It simply serves as a multiplexer or demultiplexer for sending and receiving datagrams, using ports to direct the datagrams with very thin overheads [RFC768]. Clearly, for the probing traffic transmission, UDP is a better choice than TCP, for the UDP-based probing packets are free from congestion control. However, TCP is ideal for the control channel between two ends in *Pathpair*.

6.4.3 The parameter selections

There are a number of parameters which need to be carefully considered as they can significantly influence an avail-bw tool's performance. These parameters include: the size of a probing packet, the length of a packet train and the initial packet and probing packet gaps.

- The selection of the packet size (S)

There are the following major aspects that need to be taken into account when selecting the packet size. Firstly, the packet size can not be larger than the path's MTU as to avoid fragmentation. Secondly, as Jain and Dovrolis pointed out, S can not be too small [JD02]. If the size is too small, zero-padding will cause significant changes to the probing packet size.

The effect of the packet size under PGM has been evaluated in [HS03]. The small size often leads to the underestimation of avail-bw and the large size causes overestimated outcomes [HS03]. However, such conclusions do not apply in the PRM based tools [MBG00][JD03]. In *Pathpair*, the packet size setting depends on the bottleneck capacity in the probed path. When the probed path's bottleneck capacity is low, there will be more room to adjust the sending rate if median packet size like 800 bytes is used rather than large sizes. The following table lists the packet size setting for *Pathpair* under different paths:

Table 6.3: A configuration table of packet sizes

Packet Size (S)	Bottleneck Capacity
600 Bytes	10M
800 Bytes	20M
1000 Bytes	50M
1200 Bytes	100M-500M
1500 Bytes	larger than 500M

- The selection of the probe train length (L)

First, if L is too large, the probe stream could overflow the tight link, causing packet loss in both the probe stream and cross traffic [JD02]. Second, as *Hu* et al stated, the shorter length L requires more probing phases for results to converge to find the turning point [HS03]. L is often assigned with values ranging from 60 to 100. For example, *IGI/PTR* uses 60 packet trains while *Pathload* chooses 100 packet trains. In *Pathpair*, a packet train of 100 packets is used.

6.5 The implementation of *Pathpair*

Pathpair is based on the two-end mode and consists of two separate sender (SND) and receiver (RCV) programs. It is implemented with *C++* and has been tested under Linux and FreeBSD environments.

The sending rate adjustment is realised through increasing or decreasing packet gaps. *Pathpair* uses one tenth of the corresponding bottleneck gap g_O as the initial gap g_I. Then, *Pathpair* sends out a series of packet trains to a probed path, each train with 125 UDP-based packets. The receiver (*RCV*) timestamps every receiving packet with *gettimeofday( )* function. The *RCV* will detect if the first initial sending rate is larger than avail-bw. If the *RCV* detects the turning area within the first ten K $\hat{OWD}_T$'s, it is likely that the sending rate is higher than *avail-bw*. Consequently, the *RCV* will ask the sender (*SND*) to reduce the sending rate. Specifically, the initial gap g_I will be halved at the *SND*. This process is repeated until the turning area does not appear within the first ten K $\hat{OWD}_T$ s. Then, *Pathpair* enters the turning area detection stage. Basically, if the turning area is not detected, the next sending rate will be increased, which is realised through reducing packet gaps to a small value $g_I/20$. Similar to *Pathload*, if *Pathpair* detects losses larger than 10% with a train, this train will be aborted.

There are two communication channels between the *SND* and *RCV*. One is the UDP channel used for the probing traffic transmission. To reduce the communication complexity between two ends, the sender information including initial gap, the packet length and the timestamp are directly inserted into the beginning of each probing packet. Once RCV receives a probing packet, it decodes the packet and reads the sender timestamp so as to compute a corresponding OWD. The other is a TCP based control channel. First, when the initial sending rate through this channel is higher than *avail-bw*, *Pathpair* informs the sender to adjust the sending rate. Furthermore, if the receiver

completes the avail-bw calculation, *Pathpair* will signal the sender to terminate the estimation via the TCP channel.

- Design of the architecture for *Pathpair*

Fig 6.15 presents the logical flow to enable the steps discussed above.

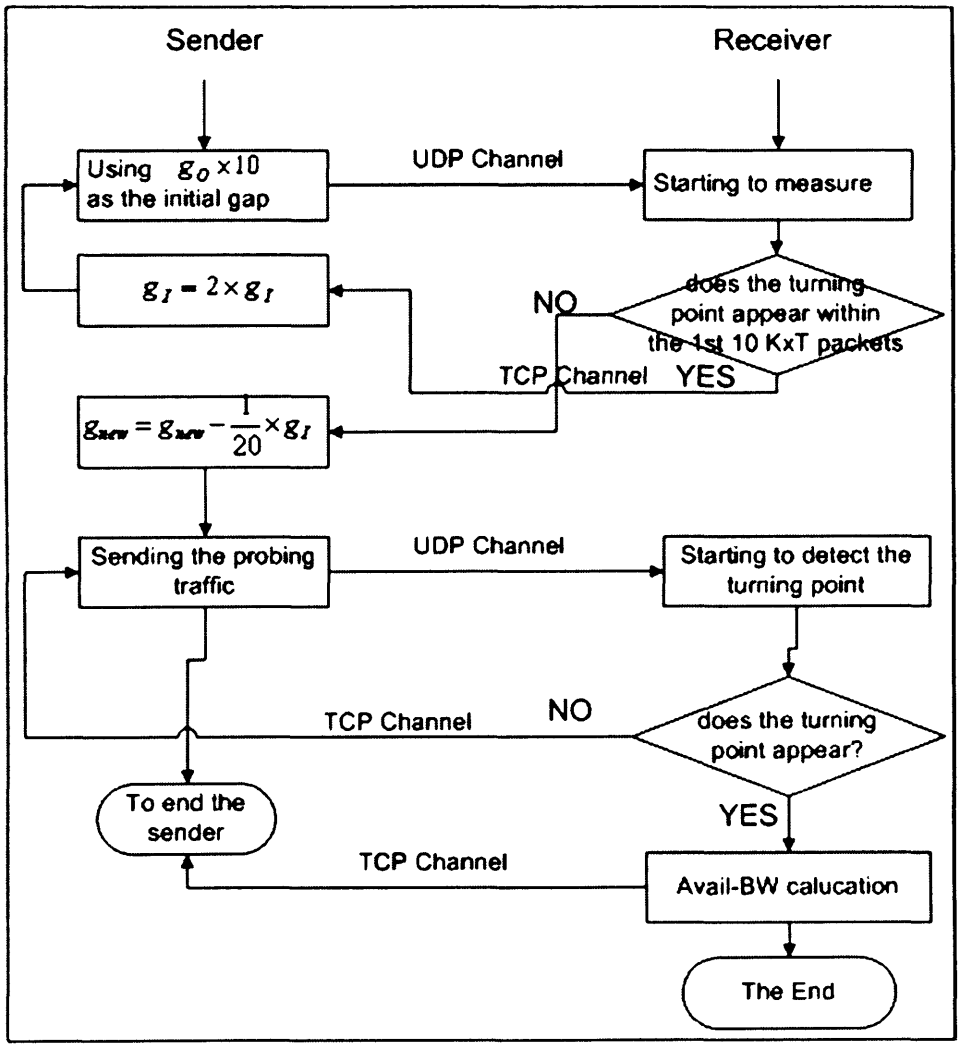


Figure 6.15: *Pathpair's* implementation architecture

- The impact of a network path's length

It is clear that the impact of the length of network path on an active probing tool should not be ignored due to different RTTs (round trip time). As RTT is proportional to the length of a network path[32], the RTT of a long network path is larger than that of a short path. In order to discuss such impacts, given a simple scenario:

> *Let L1 and L2 be the length of the 1st network path and the 2nd path, where L1>L2, let RTT1 and RTT2 be the round trip times on the paths 1 and 2, let A1 and A2 be the avail-bw values on the path 1 and path 2.*

Then the consequences include the following.

1) In Pathpair, when the destination completes the avail-bw estimation, the source will not immediately stop sending probing packets. The source will not stop until it receives the stop signal from the destination. Clearly, the faster the source receives the stop signal, the fewer probing packets will be sent out by the source.

2) In the above scenario, the amount of additional probing traffic on the 1st path is larger than that of the 2nd path as RTT1>RTT2. Thus the loading, in terms of the number of packets, is greater in the case of path 1 and, assuming A1=A2, a larger proportion of the available bandwidth is consumed during the measurement process. (Some implications of this in the real world will be briefly discussed in Chapter 7).

3) The process to calculate the amount of additional probing traffic that a source will send can be expressed by the following pseudo code. (Note that the time used to calculate additional probing traffic needs to be halved.)

[32] Under the same network conditions.

```
int AdditionalPacketCalculation()
{
 int   packetno=0  // additional probing packets
 float RTT         // the RTT of the path
 float HRTT        // the half of RTT
 float gnew        // the new time gap
 float ginit       // the initial time gap
       HRTT=RTT/2;
       for (;;) {
              if(HRTT>=125*gnew) {
                     packetno+=125;
                     HRTT=HRTT+125*gnew;
                     gnew=gnew-ginit/20;
                     }
              else {
                     packetno+=(HRTT/(125*gnew))*125;
                     break;
              }
       }
       return packetno;
}
```

6.6 Verification Experiments

This section presents a number of experiments and validation results regarding the accuracy and measurement speed of *Pathpair*. All the experiments were conducted over the Planetlab network [PLANET]. Planetlab is a group of computers available as a testbed for research in computer networking and distributed systems. It was initiated in 2002 and, by March of 2008, there are 842 nodes available accross 416 sites worldwide. The Planetlab network offers an ideal testing environment, providing a variety of Internet paths with different capacity properties for *avail-bw* evaluation. More details about Planetlab can be found in [PLANET].

Two sub-sections are included here. One is to verify the proposed AOC model while the other is to validate *Pathpair's* estimation accuracy and measurement speed.

6.6.1 Experimental results of three metrics under the AOTC model

The experiment presented in this section is conducted between two Planetlab nodes at Kent University and Queen Mary University, respectively. The bottleneck of the path from Kent to Queen Mary is around 100Mbps according to the *Pathrate's* reading.

- The calculation of three metrics and the OWD trend detection

Under the AOC model, the first step is to calculate the mean OWD for a group of 25 packets (T=25). The one way delay of a probing packet is equal to:

$$t_{RCV} - t_{SND} - t_{BIAS}$$

where t_{RCV} and t_{SND} is the packet timestamp at the *RCV* and *SND* respectively while t_{BIAS} is the time difference between two ends. t_{BIAS}, equivalent to $t_{RCV} - t_{SND}$, is calculated when the *RCV* receives the first probe packet. In this experiment, 100 packet trains (each train with 125 packets) have been sent out on the path from Kent to Queen Mary. The sending rate is realised with a reduction of the packet gap after each train as shown in Fig 6.16:

```
time_interval=PACKET_GAP;
gettimeofday(&tmp1 , NULL) ;
 t1 = tmp1.tv_sec * 1000000.0 +(double)tmp1.tv_usec;
 rc = sendto(sd, pkt_buf, strlen(pkt_buf), 0,
         (struct sockaddr *) &remoteServAddr, sizeof(remoteServAddr));
 gettimeofday(&tmp2, NULL) ;
 t2 = tmp2.tv_sec * 1000000.0 +(double)tmp2.tv_usec ;
 while( (t2 - t1) < time_interval )
  {
   gettimeofday(&tmp2, NULL) ;
   t2 = tmp2.tv_sec * 1000000.0 + (double)tmp2.tv_usec ;
  }
 PACKET_GAP - =GAP_DECREMENT;
```

Figure 6.16: The implementation of the sending rate control

Fig 6.17 plots actual OWDs against the corresponding asymptotic OWDs. For clarification note that two different averages have been used to produce this figure. Firstly, each point of the actual One Way Delays shown in the figure is calculated by Equation 6.3. The value of each point is equal to the average OWD of 25 individually measured OWD values. Secondly, each point of the Asymptotic OWD Line shown in this figure represents the accumulation of all the historic values of the average OWD's up to and including that point.

As can be seen, the OWD starts growing at around 300T and there is significant deviation between the actual OWD line and the AOT line from 300T to 400T. Before the point 300T, OWDs are rather stable as are the asymptotic OWDs.

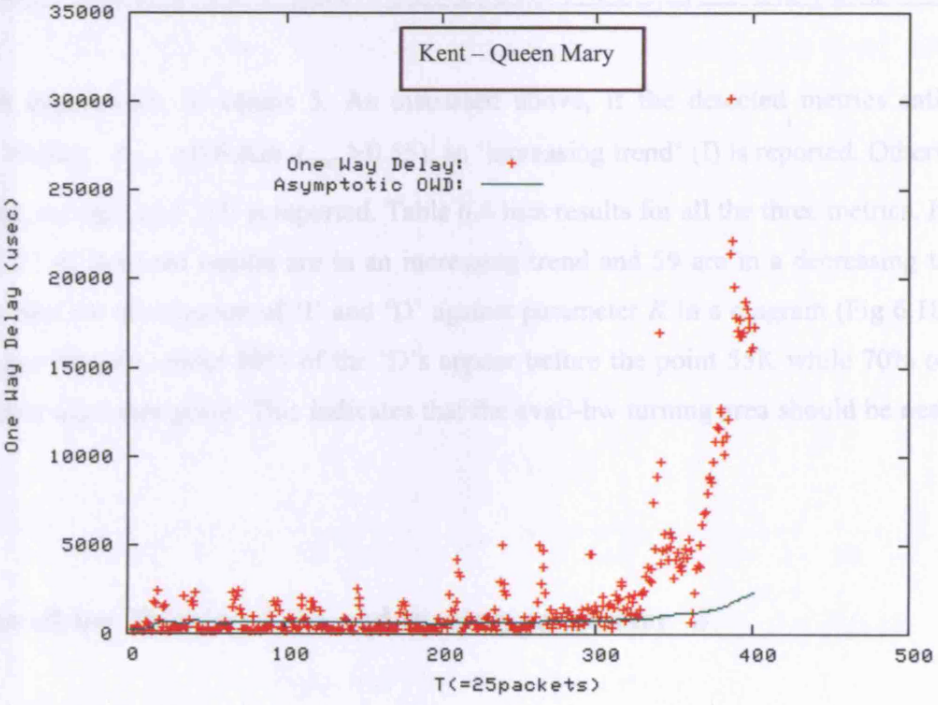


Figure 6.17: OWDs with 80 packet trains (80x5x25 packets)

Table 6.4: The outcomes of three metrics, 'I-increasing trend' and 'D-decreasing trend'

K#	A_SOT	APTC	A_CTC	I/D	K#	A_SOT	APTC	A_CTC	I/D	K#	A_SOT	APTC	A_CTC	I/D
1	0.64	0.40	0.70	D	28	-1.02	0.40	0.40	D	55	0.26	0.80	0.60	I
2	-0.52	0.20	0.20	D	29	-0.59	0.20	0.80	D	56	1.11	0.60	0.50	D
3	1.91	0.60	0.40	D	30	1.38	0.60	0.10	D	57	0.97	0.80	0.30	D
4	2.44	0.80	0.20	D	31	-2.88	0.00	0.70	D	58	0.65	0.60	0.80	I
5	0.14	0.40	0.40	D	32	-2.14	0.00	0.50	D	59	2.03	0.80	0.80	I

6	-2.37	0.00	0.70	D	33	-2.89	0.00	0.20	D	60	3.30	1.00	0.30	D
7	-0.68	0.20	0.70	D	34	-2.27	0.00	0.30	D	61	1.42	0.80	0.60	I
8	1.09	0.60	0.80	I	35	1.76	0.60	1.00	I	62	1.01	0.60	0.40	D
9	0.30	0.40	0.10	D	36	0.63	0.40	0.20	D	63	4.09	1.00	0.80	I
10	-2.91	0.00	0.50	D	37	-2.46	0.00	0.80	D	64	2.85	0.80	0.50	D
11	-3.09	0.00	0.60	D	38	-2.63	0.00	0.20	D	65	2.58	1.00	0.60	I
12	-1.71	0.00	0.70	D	39	-2.60	0.00	0.40	D	66	3.25	1.00	1.00	I
13	-0.53	0.40	0.50	D	40	-0.64	0.20	0.50	D	67	4.71	1.00	0.40	D
14	4.36	1.00	0.60	I	41	0.73	0.60	0.70	I	68	4.95	1.00	0.80	I
15	-1.81	0.20	0.10	D	42	3.76	1.00	0.80	I	69	4.80	1.00	0.60	I
16	-1.55	0.20	0.60	D	43	0.43	0.40	0.30	D	70	4.74	1.00	0.30	D
17	-2.76	0.00	0.60	D	44	0.67	0.80	0.70	I	71	4.64	1.00	0.50	D
18	-2.14	0.00	0.70	D	45	-1.78	0.00	0.40	D	72	3.41	0.80	0.30	D
19	3.83	1.00	0.50	D	46	-0.26	0.40	0.70	D	73	3.35	0.80	0.90	I
20	-2.75	0.00	0.20	D	47	0.69	0.60	0.50	D	74	4.86	1.00	0.90	I
21	0.48	0.60	0.40	D	48	2.36	0.80	0.80	I	75	4.93	1.00	0.70	I
22	0.23	0.40	0.80	D	49	1.68	0.60	0.10	D	76	4.95	1.00	0.40	D
23	0.53	0.60	0.50	D	50	-2.84	0.00	0.70	D	77	4.96	1.00	1.00	I
24	1.56	0.60	0.10	D	51	-1.25	0.20	0.60	D	78	4.97	1.00	0.00	D
25	-0.87	0.20	0.20	D	52	-0.73	0.60	0.70	D	79	4.96	1.00	0.50	D
26	-2.78	0.00	0.20	D	53	4.14	1.00	0.80	I	80	4.94	1.00	0.20	D
27	-2.92	0.00	0.50	D	54	2.34	0.80	0.30	D					

In this experiment, K equals 5. As discussed above, if the detected metrics satisfies (A_{SOT}>0 && $A_{PTC} \geq 0.6$ && $A_{CTC} \geq 0.55$), an 'increasing trend' (I) is reported. Otherwise, the 'decreasing trend' (D) is reported. Table 6.4 lists results for all the three metrics. In the Table, 21 of detected results are in an increasing trend and 59 are in a decreasing trend. Let us plot the distribution of 'I' and 'D' against parameter K in a diagram (Fig 6.18)[33] . As it can be seen, about 80% of the 'D's appear before the point 55K while 70% of the 'I's occur after that point. This indicates that the avail-bw turning area should be near the point.

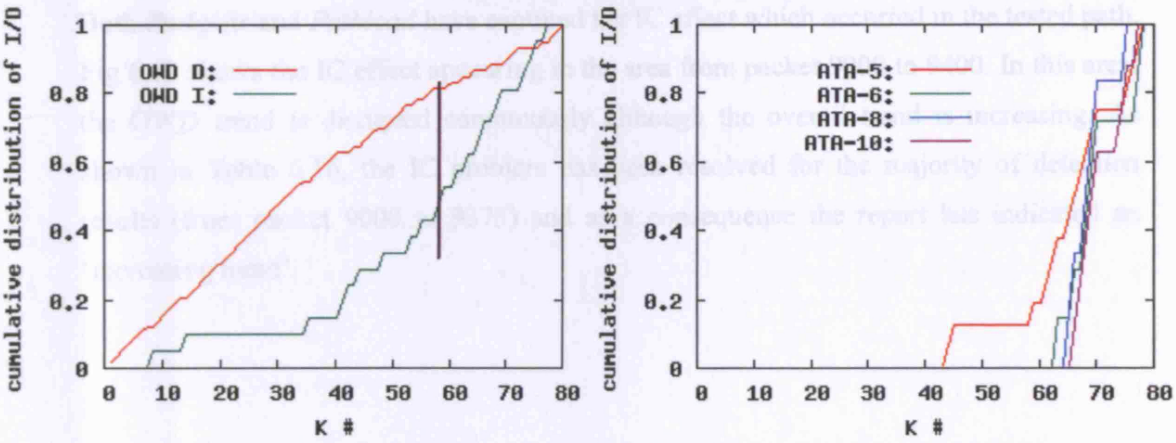
● Avail-bw Turning Area and turning parameter δ

Pathpair analyses the turning point status within a period of δ packet trains. To show the effect of the turning parameter δ, we set δ with different values: 5, 6, 8 and 10 and again plot the distribution of 'I's and 'D's in Fig. 6.18 (b). Table 6.5 lists the detected avail-bw turning area ranges in terms of parameters T, K, and δ. As can be seen from Fig

[33] The reason for choosing K as X-axis unit is that K is five times longer than T. This makes the diagram better illustrate the distribution of 'I's and 'D's.

6.18 (b) and Table 6.5, the majority of the detected turning areas are distributed after the point K=57K. In addition, Fig 6.18 (b) indicates that if δ is too small, the detected turning area appears earlier than the actual turning area. For example, when δ=5, the first detected ATA position is between 40K to 44K and the ATA positions appear between 200T-220T (Table 6.5). This result is incorrect if we look at the Figure 6.18.

In the cases where δ equals 6, 8 and 10, the related first ATA ranges are quite close to one another i.e. in the rage K=57K to 66K. According to the fast growing value in A_{SOT} and the large value in A_{PTC} after the point 57K (Table 6.4), it shows that such an ATA range is very close to the actual ATA. Based on this detected ATA, *Pathpair*'s reading on avail-bw is 87.16Mbps while *Pathload's* estimate is 86Mbps. Note that the *Pathload* measurement was started shortly after *Pathpair's* measurement – within less than 1 minutes. More details about the avail-bw measurement results are given in the following section 6.6.2.



(a) The I/D distribution (b) The I/D distribution with δ=5,6,8,10

Figure 6.18: The distribution of I/D

Table 6.5 – The ATA range with different δ values

δ #	5	6	8	10
K #	40-44	58-63	58-65	57-66
T #	200-220	290-315	290-325	285-330

- **The turning parameters T and K**

As discussed above, the correct OWD trend detection is based on the setting of turning parameters that satisfy: T=25 and K=5. It should be noted that such settings comply with those in *Pathload*. In *Pathload*, the trend detection of OWDs is made within every packet stream. The length of each stream equals 100 packets. The 100-packet length is very close to the length ($K \times T = 5 \times 25 = 125$ packets) used in *Pathpair*. The reason for using such a length has been explained in [JD02]:

> *"Pathload uses K=100 packets, because this stream length rarely causes packet losses, while it provides an adequate number of OWD measurements to detect an increasing trend."*

- **IC effect**

Both *Pathpair* and *Pathload* have captured the IC effect which occurred in the tested path. Fig 6.19 shows the IC effect appearing in the area from packet 9000 to 9400. In this area, the OWD trend is disrupted continuously although the overall trend is increasing. As shown in Table 6.16, the IC problem has been resolved for the majority of detection results (from packet 9000 to 9375) and as a consequence the report has indicated an 'increasing trend'.

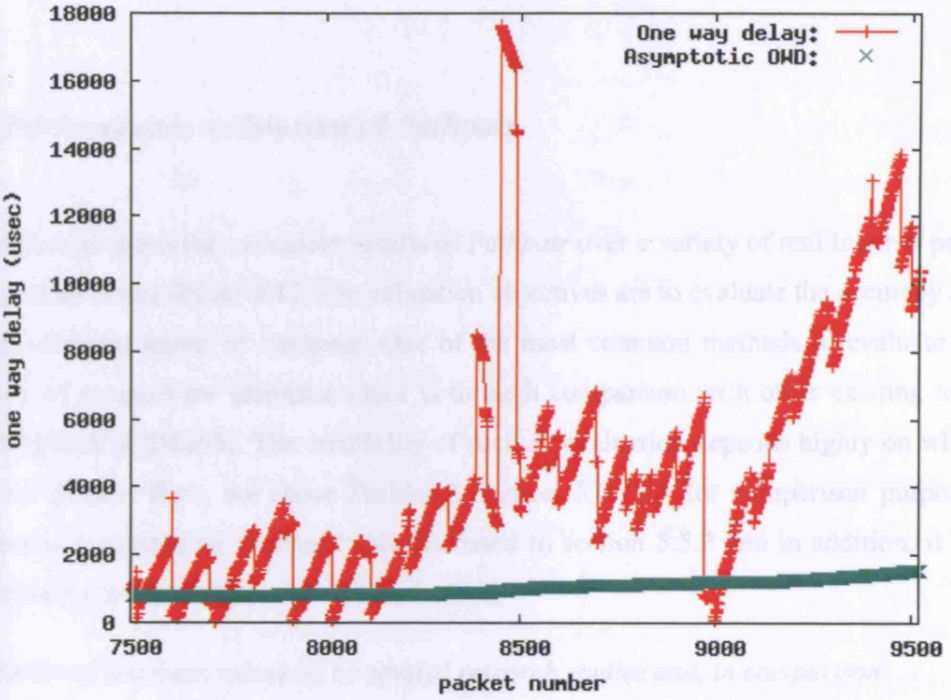


Figure 6.19: The captured IC effect

Table 6.6: The IC effect

packet #	T#	K #	A_SOT	A_PTC	A_CTC	Increasing /Decreasing
9000-9125	365-370	73	3.35	0.80	0.90	I
9125-9250	370-375	74	4.86	1.00	0.90	I
9250-9375	375-380	75	4.93	1.00	0.70	I

6.6.2 Performance validation of *Pathpair*

This section presents the validation results of *Pathpair* over a variety of real Internet paths on PlanetLab nodes [PLANET]. The validation objectives are to evaluate the accuracy and the measurement speed of *Pathpair*. One of the most common methods to evaluate the accuracy of an avail-bw estimation tool is through comparison with other existing tools [SMH05][SKK03][HS03]. The credibility of such an evaluation depends highly on which tools are chosen. Here, we chose *Pathload, Spruce, IGI/PTR* for comparison purposes. The reason for choosing *Pathload* was discussed in section 5.5.2 and in addition, it has been reported in [AAP06]:

> *"Pathload has been validated by several research studies and, in comparison with other avail-bw estimation tools, it was shown to be the most accurate"*

Second, as both *Pathload* and *Spruce* has been evaluated by MRTG [ORR] , which is one of the most reliable evaluation methods, the comparison between *Pathpair* and these two tools can be regarded as an indirect comparison method with MRTG. Moreover, the evaluation has been carried out over a variety of Internet paths that having different ranges of bottlenecks from low to high in terms of capacity. In this section, the *IGI/PTR* tool is used for the measurement speed comparison as it has been claimed to be one of the fastest avail-bw estimation tools available [HS03].

- **The testing of the paths with bottleneck capacities up to 100Mbps**

1) The path between PlanetLab nodes: planetlab.kent.ac.uk - planetlab.cs.uchicago.edu

The first tested path is between two Planetlab nodes, Kent University and Chicago University, and relevant routes for this path are given in Appendix E. The bottleneck capacity of this path (in the direction from Kent to Chicago) is approximately 80Mbps according to *Pathrate's* reading. Table 6.7 lists all the measured results from the four tools.

There are 10 groups of measurement results in total in this table. *Pathload's* avail-bw estimate is calculated as the average of the high and low estimates. The avail-bw estimate for *PTR/IGI* only uses the suggested value for the comparison (marked with * in table). Table 6.8 (a) ranks the variation between *Pathpair* and three other tools. As shown in this table, the avail-bw estimates from *Pathpair* are very close to both *Pathload* and *Spruce*, for about 50% of the runs is within 5% difference and another 40% of the runs is within 15% difference. However, *PTR*'s estimates have shown the larger deviation to the other three tools with only 40% of runs within 15% variation. Fig 6.20 (a-c) shows the variations between *Pathpair* and the other three tools. The data used to draw the error bars is taken from Table 6.8 (b); this Table includes the mean and twice the standard error, for the data in Table 6.7(twice the standard error is used to obtain 95% confidence limits). From the difference between error bars, it can be seen that there are around 3Mbps (about 4.7%) mean difference between Pathload, Spruce and Pathpair. Although PTR's mean value is the closest to Pathpair's, its variability is the largest among all the tools. Table 6.7 also presents the measurement time of the four tools; it is clear that *Pathpair* needs the least time with an average of 2.5 seconds. *PTR/IGI* is the second fastest tool with the average of time 5.92 seconds. The average measurement time for *Spruce* is 12.3 seconds while *Pathload* needs the longest time for each run requiring 32.03 seconds on average.

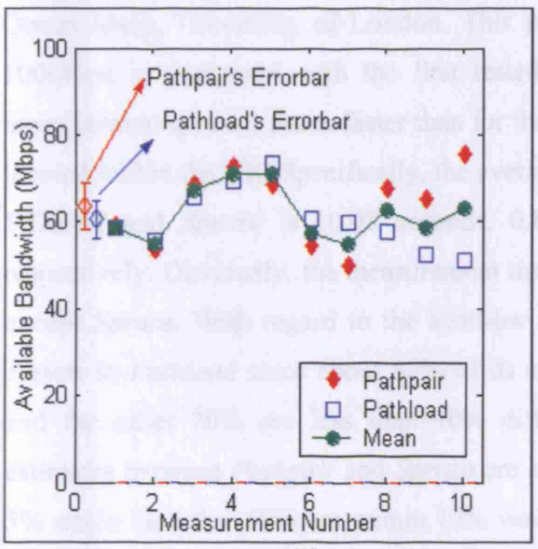
Table 6.7: Kent Planetlab size - Planetlab1.cs.uchicago.edu

(Note the measurement is conducted with the following cycle. Firstly, the measurement is run with the order *Pathload*, *Pathpair*, *PTR/IGI* and then *Spruce*; this is called one measurement group. Each group of measurements took 4 minutes ie it took a total of 40 minutes to run 10 groups of measurements. Such a cycle is also used in the rest of tests (eg Table 6.9 and Table 6.11))

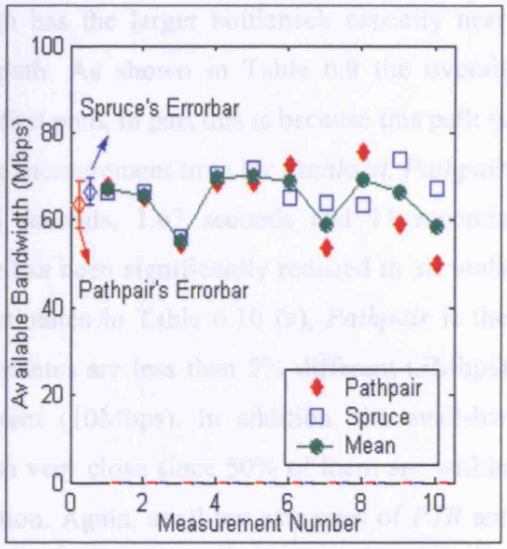
No	*Pathload* (Mbps)	Time (sec)	*Pathpair* (Mbps)	Time (sec)	*PTR* (Mbps)	*IGI* (Mbps)	time (sec)	*Spruce* (Mbps)	Time (sec)
1	73.25	32.32	68.45	1.9	92.672 (*)	162.306	6.721	70.462	11
2	69.35	31.78	72.51	2.8	43.353 (*)	174.432	4.427	65.132	12
3	52.20	37.40	65	2.1	50.958 (*)	464.976	2.826	66.384	11
4	51.02	29.73	75.23	2.5	85.788 (*)	745.936	2.391	63.381	11
5	58.40	36.94	58.94	2.6	45.038 (*)	147.885	3.267	73.755	14
6	65.7	27.99	68.52	2.8	57.115 (*)	135.542	6.090	71.524	13
7	57.60	28.24	67.66	2.6	69.227 (*)	257.236	9.549	66.337	14
8	55.60	38.45	53.69	2.7	75.739 (*)	159.495	3.690	63.692	11
9	60.5	29.10	54.31	2.56	44.796 (*)	125.476	2.385	56.055	14
10	59.70	28.38	49.82	2.5	72.273 (*)	116.907	17.873	67.214	12

Table 6.8 (a): The avail-bw estimate difference ranking (Mbps)

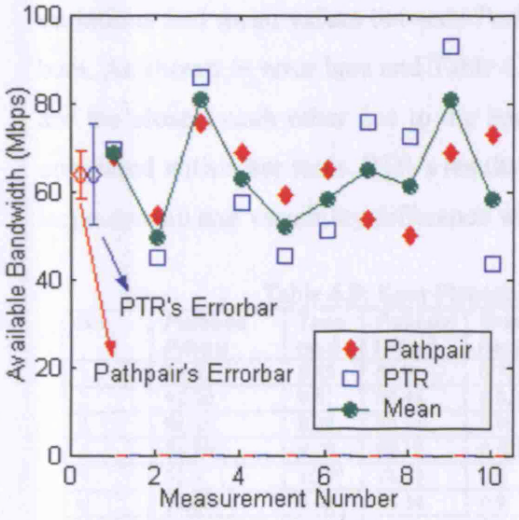
Ranking	1	2	3	4	5	6	7	8	9	10
Pathload	0.54	1.92	3.01	3.15	3.71	4.79	9.88	10.06	12.80	24.23
Spruce	1.32	1.38	1.74	2.01	3.01	7.37	10.01	11.84	14.81	17.39
PTR	1.56	9.51	10.5	11.4	13.9	14.03	22.05	22.45	24.22	29.15



(a)



(b)



(c)

Table 6.8(b):The mean and standar error

Tools	mean	$\frac{2\sigma}{\sqrt{n}}$
Pathpair	63.4130	5.4836
Pathload	60.3320	4.5405
Spruce	66.3936	3.1453
PTR	63.6959	11.3560

(Note that the ' Pair Mean' shown in figure a-c is the mean of successive measurements using two tools within the same measurement group. While the mean shown in error bars (or table 6.8-b) refers to the [mean +- 2x(standard error)] of the 10 repeat test data values for of each individual tool, from Table 6.7.)

Figure 6.20: The comparison results between *Pathpair* and other tools

2) The path between planetlab.kent.ac.uk and planetlab.nrl.dcs.qmul.ac.uk

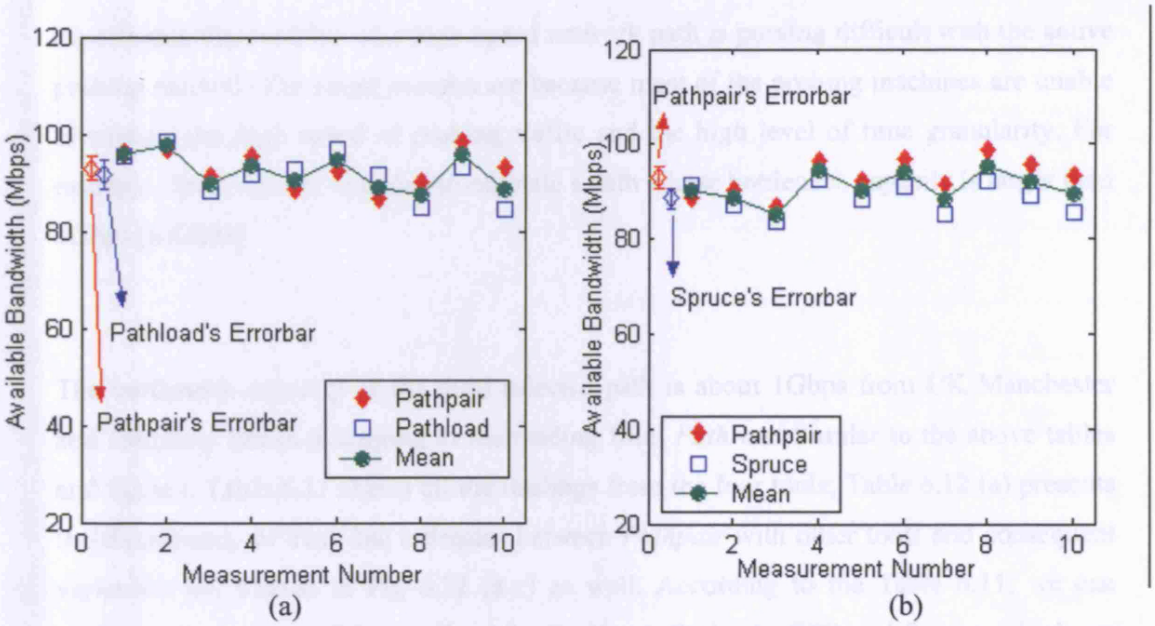
The second chosen path is between two Planetlab nodes within UK: Kent University and Queen Mary, University of London. This path has the larger bottleneck capacity near 100Mbps in compared with the first tested path. As shown in Table 6.9 the overall measurement speed is much faster than for the first path, in part this is because this path is located within the UK. Specifically, the average measurement time for *Pathload, Pathpair, PTR/IGI* and *Spruce* is 10.08 seconds, 0.86 seconds, 1.67 seconds and 11 seconds respectively. Obviously, the measurement time has been significantly reduced in all tools except *Spruce*. With regard to the avail-bw estimates in Table 6.10 (a), *Pathpair* is the closest to *Pathload* since about 80% of its estimates are less than 5% different (5Mbps) and the other 20% are less than 10% different (10Mbps). In addition, the avail-bw estimates between *Pathpair* and *Spruce* are also very close since 50% of them are within 5% while the other 50% are within 10% variation. Again, avail-bw estimates of *PTR* are the least close to the other three tools. Fig 6.21 (a-c) plots related avail-bw estimate variations and mean values between *Pathpair* and other tools as well as the relevant error bars. As shown in error bars and Table 6.10 (b), the results between Pathpair and Pathload are the closest each other due to the least mean and variability difference between them compared with other tools. PTR's results agree the least with other tools' as PTR holds the largest mean and variability difference with other tools'.

Table 6.9: Kent Planetlab site – QueenMary Planetlab site

No	*Pathload* (Mbps)	Time (sec)	*Pathpair* (Mbps)	Time (sec)	*PTR* (Mbps)	*IGI* (Mbps)	time (sec)	*Spruce* (Mbps)	Time (sec)
1	84.87	8.45	89.9	0.727	124.198 *	81.219	1.674	87.07	12
2	92.90	9.41	98.45	0.829	89.152 *	85.445	2.456	92.044	10
3	96.65	8.38	92.14	0.99	102.203 *	88.074	1.166	88.201	10
4	84.20	9.09	93.12	0.525	124.795 *	222.990	2.893	85.253	13
5	92.5	10.77	88.61	0.5	102.804 *	80.781	0.400	91.396	11
6	91.60	9.92	95.34	0.9	131.929 *	150.242	0.784	88.850	11
7	88.1	13.88	91.1	0.41	74.874 *	73.697	2.979	84.884	10
8	91.45	9.25	86.45	0.392	72.466 *	74.536	0.488	83.341	10
9	95.20	13.53	96.08	1.2	108.858 *	76.148	1.451	92.521	12
10	98.15	8.10	96.77	2.1	107.89 *	76.57	2.41	91.034	11

Table 6.10 (a): The avail-bw estimate difference ranking (Mbps)

Ranking	1	2	3	4	5	6	7	8	9	10
Pathload	0.88	1.20	1.38	3.00	3.39	3.74	4.81	5.00	5.55	8.92
Spruce	2.79	2.83	3.11	3.56	3.94	5.74	6.22	6.41	6.49	7.87
PTR	9.30	10.06	11.12	12.78	13.98	14.19	16.23	31.67	34.30	36.59



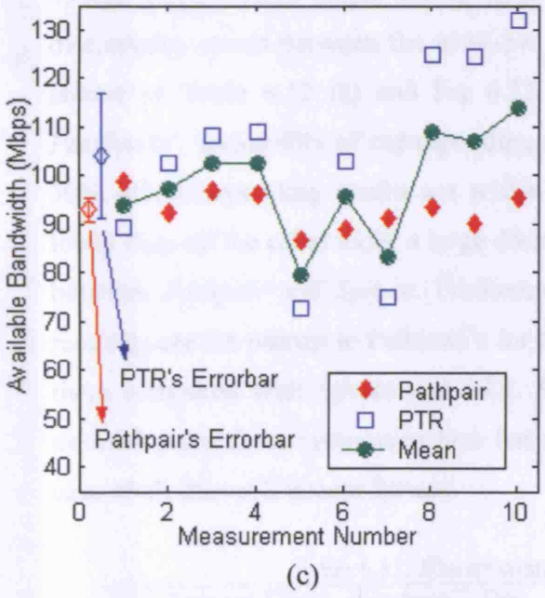


Table 6.10(b): The mean and standard error

Tools	mean	$\frac{2\sigma}{\sqrt{n}}$
Pathpair	92.7960	2.4459
Pathload	91.5620	2.9528
Spruce	88.4594	2.0657
PTR	103.9169	12.8702

Figure 6.21: The comparison results between *Pathpair* and other tools

- **In a High Speed environment**

To estimate the avail-bw of a high-speed network path is proving difficult with the active probing method. The major reasons are because most of the probing machines are unable to support the high speed of probing traffic and the high level of time granularity. For example, *Spruce* is not intended to estimate a path whose bottleneck capacity is larger than 1Gbps [SKK03].

The bottleneck capacity of the third selected path is about 1Gbps from UK Manchester and Germany Berlin according to the reading from *Pathrate.* Similar to the above tables and figures, Table 6.11 shows all the readings from the four tools; Table 6.12 (a) presents the discrepancy of avail-bw estimates between *Pathpair* with other tools and consequent variations are plotted in Fig 6.22 (a-c) as well. According to the Table 6.11, we can compute the average of the readings for *Pathload, Pathpair, PTR and Spruce*, which are 499.16 Mbps, 502.12 Mbps, 618.48 Mbps and 292.75 Mbps respectively. As such, a large discrepancy exists between the avail-bw estimates from *Pathload*, *PTR* and *Spruce*. As shown in Table 6.12 (a) and Fig 6.22, *Pathpair's* readings agree most closely with *Pathloads';* about 60% of corresponding results show less than 5% variation and another 30% of corresponding results are within about 10%. As *Spruce's* readings are notably lower than all the other tools, a large discrepancy (>15%) of avail-bw estimates is evident between *Pathpair* and *Spruce.* Furthermore, the error bars also suggest that Pathpair's readings are the nearest to Pathload's for the least mean and variability difference between them compared with Spruce and PTR. Since the testing is conducted on a high speed network path, the measurement time has reduced again (compared with Table 6.7) in the case of all the tools except *Spruce.*

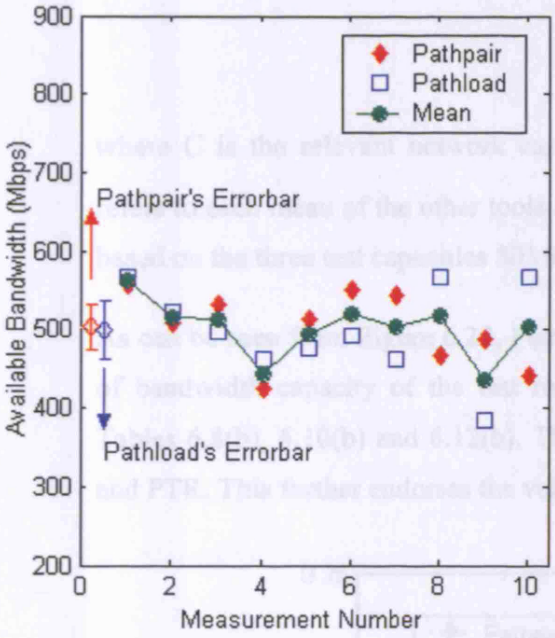
Table 6.11: Planet.manchester.ac.uk - Planet.zib.de

No	*Pathload* (Mbps)	Time (sec)	*Pathpair* (Mbps)	Time (sec)	*PTR* (Mbps)	*IGI* (Mbps)	time (sec)	*Spruce* (Mbps)	Time (sec)
1	521.74	10.01	507.33	0.51	859.623 *	849.747	2.012	356.237	14
2	384.26	4.82	487	0.95	581.632 *	858.179	0.695	268.198	12
3	566.86	7.39	441.04	0.48	874.431 *	863.503	0.969	229.079	13
4	473.92	5.54	513.5	0.88	487.488 *	362.789	0.494	295.558	12
5	566.86	7.07	466.33	1.15	729.616 *	896.449	2.850	342.004	11
6	566.86	7.02	558	0.82	524.747 *	572.640	1.487	385.270	12
7	496.33	5.31	531	1.09	498.647 *	888.294	0.525	352.921	12
8	461.54	4.14	543	1.05	849.097 *	874.168	0.478	191.640	11

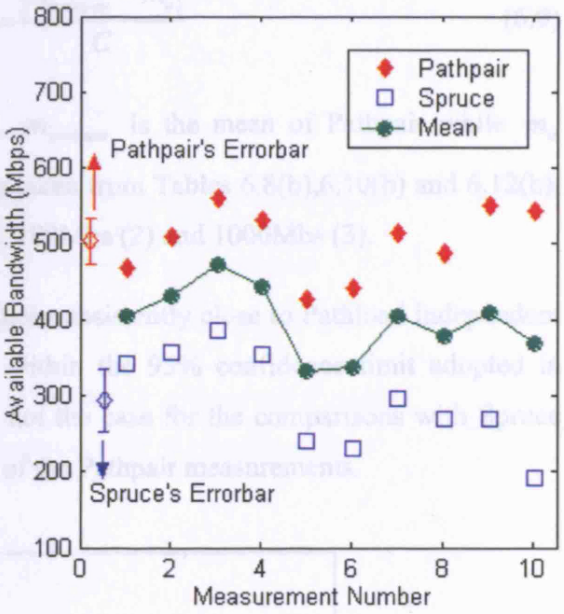
9	461.54	5.40	424.66	0.81	623.583 *	850.733	0.470	239.417	12
10	491.67	9.97	549.33	0.88	155.912 *	883.587	2.10	267.262	12

Table 6.12 (a): The avail-bw estimate difference ranking (Mbps)

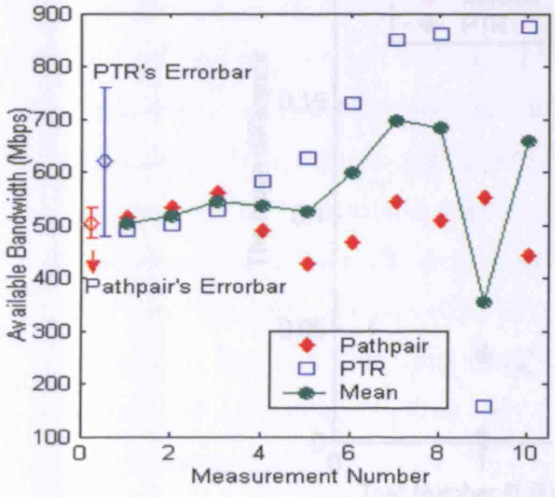
Ranking	1	2	3	4	5	6	7	8	9	10
Pathload	8.86	14.41	34.67	36.88	39.58	57.66	81.46	100.53	102.74	125.82
Spruce	124.33	151.09	172.73	178.08	185.24	211.96	217.94	218.80	282.07	351.36
PTR	26.01	32.35	33.25	94.63	198.92	263.29	306.10	352.29	393.42	433.39



(a)



(b)



(c)

Table 6.12(b): The mean and standar error

Tools	mean	$\frac{2\sigma}{\sqrt{n}}$
Pathpair	502.1190	29.2526
Pathload	499.1580	37.1145
Spruce	292.7586	40.5622
PTR	618.4776	140.6104

Figure 6.22: The comparison results between *Pathpair* and other tools

- **The mean difference summary**

In Figure 6.23, the absolute percentage difference of the means ΔM between Pathpair and three other tools has been plotted against the test number. The percentage difference of the means is defined as:

$$\Delta M = \frac{\left| m_{pathpair} - m_o \right|}{C} \tag{6.9}$$

where C is the relevant network capacity, $m_{pathpair}$ is the mean of Pathpair while m_o refers to each mean of the other tools and is taken from Tables 6.8(b),6.10(b) and 6.12(b), based on the three test capacities 80Mbs (1), 100Mbs (2) and 1000Mbs (3).

As can be seen from Figure 6.23, Pathpair lies consistently close to Pathload independent of bandwidth capacity of the test routes, within the 95% confidence limit adopted in Tables 6.8(b), 6.10(b) and 6.12(b). This is not the case for the comparisons with Spruce and PTR. This further endorses the validity of the Pathpair measurements.

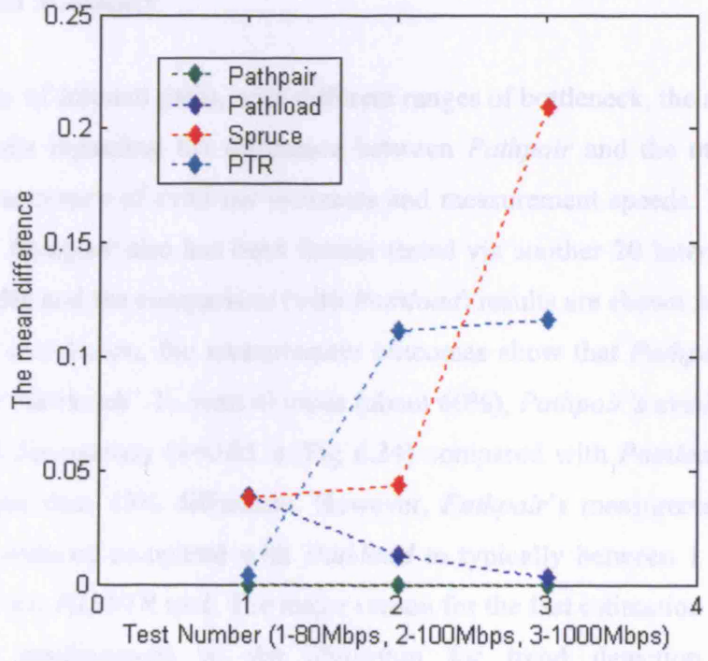


Figure 6.23: The mean difference between Pathpair and other tools (y-axis : ΔM)

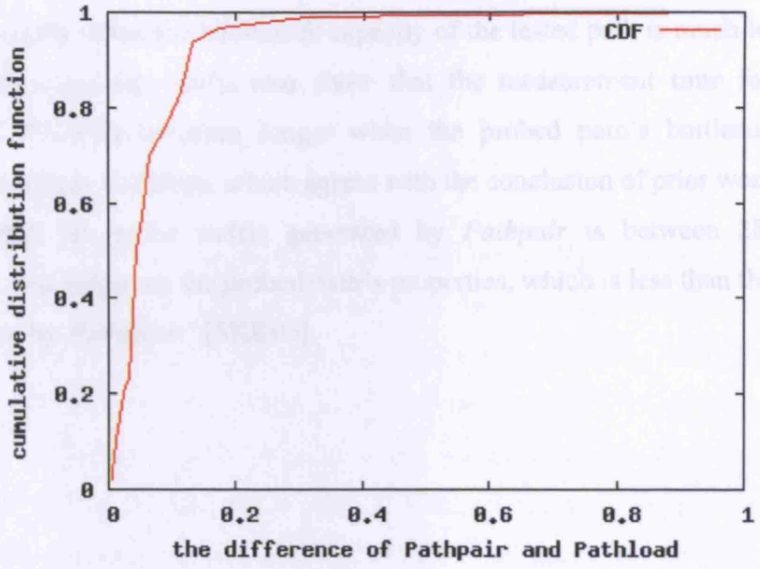


Figure 6.24: The comparison results between *Pathpair* and *Pathload*

(X-axis: the estimation difference between *Pathpair* and *Pathload*
Y-axis: the cumulative distribution of the difference.)

- **Validation Summary**

Over a variety of Internet paths, with different ranges of bottleneck, the above experiments give the details regarding the difference between *Pathpair* and the other three tools in terms of the accuracy of avail-bw estimates and measurement speeds. In the above three experiments, *Pathpair* also has been further tested via another 20 Internet paths between Planetlab nodes and the comparison (with *Pathload*) results are shown in Fig 6.24. Similar to the above conclusion, the measurement outcomes show that *Pathpair*'s estimates are the closest to *Pathloads'*. In most of cases (about 60%), *Pathpair's* avail-bw reading show less than 5% discrepancy (x=0.05 in Fig 6.24) compared with *Pathloads'* while another 30% show less than 10% difference. However, *Pathpair*'s measurement time has been significantly reduced compared with *Pathload* to typically between 1 to 3 seconds, the same level as the *IGI/PTR* tool. The major reason for the fast estimation speed of *Pathpair* arises from improvement to the algorithm for trend detection of OWDs and implementations of the avail-bw turning area. *Pathpair's* estimates can be close to

Spruces', especially when the bottleneck capacity of the tested path is much less than 500 Mbps. The experimental results also show that the measurement time for *Pathload, Pathpair and IGI/PTR* becomes longer when the probed path's bottleneck capacity decreases to less than 100Mbps, which agrees with the conclusion of prior work [SMH05]. It is noted that the probe traffic generated by *Pathpair* is between 2M-5MB per measurement, depending on the probed path's properties, which is less than the 2.5-10MB range produced by *Pathloads'* [SKK03].

6.7 Summary and future work

This chapter goes further to address several of the remaining problems of OWD trend detection. Specifically, the problems arise from the burst cross-traffic, the changing performance at the probed machines and the interrupt coalescence. As a consequence of these factors, OWDs often change irregularly and dynamically. The dynamics of OWDs clearly are an issue for OWD trend detection realised by the comparison between successive OWDs. To deal with this issue, this chapter proposed a novel solution – an asymptotic OWD comparison model for OWD trend detection. The principle behind the AOC model is based on two aspects. Firstly, an asymptotic OWD line is more stable than the actual OWD line; therefore, the asymptotic OWD line has been introduced to simulate the level line of the OWDs as discussed in section 6.2. Secondly, after the turning point, the trend of the asymptotic OWDs grows much slowly than is the case for the actual OWDs. Consequently, by monitoring the difference between asymptotic OWDs and actual OWDs, the avail-bw turning point can be detected. Based on this model, an *avail-bw* estimation tool called *Pathpair* has been developed.

The *Asymptotic OWD Comparison* (AOC) algorithm is realised by analysing three statistical metrics: A_{SOT}, A_{PTC} and A_{CTC}. The input to calculate the three metrics is not a real OWD value but a transformed value ranging from (-1,1), in order to reduce the excessive range of the OWDs. The A_{SOT} and A_{PTC} metrics indicate whether current OWDs are above the asymptotic OWD line or not; while the A_{CTC} metric shows the rising strength of current OWDs. Through a comparison with PCT/PDT based algorithms, the merits of the proposed AOC algorithms have been well demonstrated in a number of typical practical cases. The process to detect an *avail-bw* detection turning point is controlled by three elaborate parameters: T, K and δ. T is used to remove outliers of

OWDs; K serves for the trend detection within a group of OWDs while δ is used for the avail-bw turning area detection.

Pathpair is developed under the two-end mode, including one UDP channel to transmit probing packet and the other TCP channel is used for secure communication between the two ends. The initial gap setting is judged according to the bottleneck capacity, which can be obtained from *Pathrate's* reading.

All the experiments are conducted over the real Internet paths in Planetlab. The experiments mainly consist of two parts. The first part examines the proposed AOC algorithms and presents the related detection results. The testing results show when T=25, K=5 and $\delta \in [6,10]$[34], *Pathpair* yields close avail-bw estimates in compared with *Pathload*. The insight to such kind of parameter setting complies with *Pathload*'s setting. In addition, the proposed AOC algorithm also produces the correct detection result under the presence of the *interrupt coalescence*. The second part presents the *Pathpair's* validation results including accuracy and measurement speed through the comparisons with *Pathload, Spruce and IGI/PTR*. The experiments are conducted over 20 different Internet paths with a wide spectrum of avail-bw values. Experimental results show that 60% of *Pathpair's* avail-bw reading are less than 5% discrepancy with *Pathloads'* and 30% of that are less than 10% different, while its measurement speed (1-3 seconds) has been significantly improved to the same level as *IGI/PTR*'s, which is currently one of the fastest avail-bw tools. The major reason for the fast estimation speed of *Pathpair* is due to the algorithm improvement on the trend detection of OWDs and the avail-bw turning area identification.

[34] It has been set to a fixed number 10 in *Pathpair*. (i.e. *Pathload* uses 12 packet streams for the ATP detection).

Chapter 7

Conclusion and Future Work

7.1 Summary and major contributions

This thesis explored two important network measurement schemes in IP networks: an automatic traffic classifier and a fast available bandwidth estimation tool using the active probing approach.

7.1.1 The research in Traffic Classification

The motivation for building an automatic traffic classifier comes from the fact that recent years have seen fast growth in the number and variety of network applications that are not based on registered or well-known port numbers. In this thesis, the goal of developing an ***Automatic*** and ***Accurate*** classification scheme has been studied as summarised below.

For the purpose of automatic classification, first of all, the IP flow profile concept has been proposed and the associated model using five IP header based contexts has been presented in this thesis. This research has shown that the key statistical features of each context, in the IP flow profile, follows a Gaussian distribution. In addition, this thesis has shown that the Kohonen Neural Network (KNN) is a good candidate to be used for automatically producing IP flow profile maps since its core learning algorithm is Gaussian-based and it has an inherently powerful capability to transform a high-dimensional IP flow profile into a low dimensional map in a self-organising and

unsupervised manner. The experimental results have illustrated that the KNN was an effective and efficient means to separate varying network traffic classes into different areas in the produced IP flow profile map. This thesis also has discussed that the classification cost could be significantly reduced by only considering long-lived flows, if the classification is used for traffic engineering purposes, since the short-lived flows can often be omitted. Furthermore, the ontological view of the relationship between the proposed classification concept and other concepts has been presented.

To achieve accurate traffic classification results, this research firstly has addressed the feature selection issue. The purpose of feature selection in this research is to make the produced traffic patterns in an IP flow profile map as tight as possible. This research has shown that PCA is an effective technique in the feature selection, for it is able to quickly identify which inputs contribute the least to the principal components in terms of the associated variability. Experimental results have demonstrated that tight patterns have been produced according to the input selection using PCA.

In addition, this research has tackled the overlap issue. LDA has been investigated and experimental results have shown that it was a fast and efficient way to deal with the partial overlap problem. For the selected five categories, the classification accuracy has already reached as high as 91.86% with the use of PCA and LDA. The reason for the 8% misclassification is that whereas the inputs produced by PCA may be optimal as a whole, they may not be optimal for an individual application case. Therefore, the second (alternative) map is proposed to deal with the complete overlap problem in this thesis. Through constructing a second KNN map, taking *Skype* and *Pool* as examples, the classification results can reach 100% accuracy.

The speed of the proposed classifier is another promising feature. The time spent on KNN training and PCA is only required once during the offline period while the online

classification time needed by KNN and LDA is at the microsecond level. The proposed classifier is potentially an early classification solution for it often can give classification results much before a flow is ended.

7.1.2 The research in Avail-bw Estimation

To begin with, the research has investigated related concepts and techniques in the active avail-bw estimation area during the past decade. The key aspects of the Probe Rate Model and the Probe Gap Model and corresponding techniques used in current major avail-bw estimation tools have all been discussed.

The motivation of developing a fast OWD based avail-bw estimation tool has been addressed. It has been shown that OWDs often change irregularly and dynamically because of the burst cross-traffic, the changing performance at the probed machines and the interrupt coalescence. Hence, this thesis proposed an asymptotic OWD comparison (AOC) model for OWD trend detection. It has been shown that the proposed AOC model effectively enhanced the trend detection reliability and robustness of OWDs for avail-bw estimation compared with *Pathload*'s PCT/PDT. Based on this model, an *avail-bw* estimation tool called *Pathpair* has been developed. *Pathpair* was developed under the two-end mode, including one UDP channel to transmit probing packet and the other TCP channel was used for secure communication between the two ends. The initial gap setting is judged according to the bottleneck capacity.

The validation of *Pathpair* has been conducted over 20 different Internet paths with a wide spectrum of avail-bw values. Firstly, three turning parameters have been evaluated. The experimental results showed when T=25, K=5 and $\delta \in [6,10]$[35], *Pathpair* yielded close avail-bw estimates compared with *Pathload*. It has been shown that such settings

[35] It has been set to a fixed number 10 in *Pathpair*. (i.e. *Pathload* uses 12 packet streams for the ATP detection).

comply with *Pathload's* setting. In addition to that, experimental results also showed that the correct detection results have been produced under the presence of *interrupt coalescence.*

Pathpair's evaluation, including accuracy and measurement speed, were carried out through comparisons with *Pathload, Spruce and IGI/PTR.* Experimental results showed that 60% of *Pathpair's* avail-bw reading indicated less than 5% discrepancy compared with *Pathloads'* and 30% indicated less than 10% difference. *Pathpair's* measurement speed (1-3 seconds) shows a significant improvement compared with *Pathload* and reaches the same level as *IGI/PTR*'s, which is currently one of the fastest avail-bw tools. The major reason for the fast estimation speed of *Pathpair* is because of the algorithm improvement for the trend detection of OWDs.

7.2 Major Contributions

7.2.1 Contributions in traffic classification

IP flow profile concept

The proposed automatic classifier is based on the IP flow profile concept, defined as the high-level description and representation of traffic features at the flow level. The underlying principle of using the flow profile concept is due to the fact that each network application has its own flow profile feature, which is unique and independent from one network traffic class to another. In addition to that, the flow profile of a network class is robust and stays unchanged no matter whether the application traffic has been wrapped into other protocols or has been encrypted. As discussed in section 2.3, the IP flow profile concept also aids the organisation of the rich context input data. Specifically, an IP flow profile model using five IP-header based contexts has been proposed. Furthermore, based

on the IP flow profile concept, the classification cost can be significantly reduced when only long-lived flows are considered rather than all network flows.

The application of Kohonen Neural Network to produce IP flow profile map

This research has investigated the criteria for algorithm selection in order to produce an IP flow profile map, mainly including two aspects: one is the ability to match the Gaussian distribution feature of each context input while the other is the ability to deal with *'the curse of dimensionality'* issue. This thesis showed that KNN met both criteria. In addition to that, KNN's learning ability to input data is achieved through the self-organising and unsupervised features. This brings the automatic learning ability into the proposed classifier.

The application of KNN for traffic classification has been evaluated through a series of experiments with real network traffic data. The IP flow profile maps (Fig 3.24 and Fig 4.7) produced by KNN demonstrated KNN's powerful classification ability to detect varying network traffic categories. Even applications holding analogous traffic profiles like Realplayer and Mediaplayer have been effectively separated by KNNs. The results also show that KNN's classification time is very fast (usually from 3 seconds to 10 seconds) to classify a network traffic class.

Classification accuracy

Another major contribution is classification accuracy. The idea has been proposed that classification accuracy could be achieved through three steps as shown in Fig 7.1. The first step is feature selection, which allows the produced patterns to be as tight as possible; tight patterns lead to less overlaps. PCA has been investigated and evaluated for the feature selection. The second step is to deal with the partial overlap problem. This is because the shapes of the produced patterns are arbitrary. This research has shown that the partial overlap issues have been minimised through the projected LDA results. In addition, it is easy to read different traffic patterns with the use of LDA. The third step is to tackle

the complete overlap issue. The thesis has proposed the use of an alternative KNN map to deal with this issue.

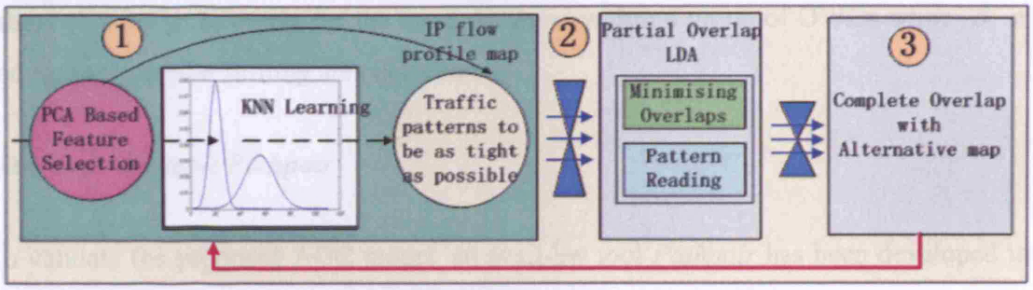


Figure 7.1: The three steps regarding classification accuracy.

7.2.2 Contributions in Avail-bw Estimation

The Asymptotic OWD Comparison model

This work proposed the *Asymptotic OWD Comparison* model to deal with OWD dynamics issue. The principle behind the AOC model is based on two aspects. Firstly, an asymptotic OWD line is more stable than the actual OWD line; therefore, the asymptotic OWD line has been introduced to simulate the level line of the OWDs, as discussed in section 6.2. Secondly, after the turning point, the trend of the asymptotic OWDs grows much slowly than is the case for the actual OWDs. Consequently, by monitoring the difference between asymptotic OWDs and actual OWDs, the avail-bw turning point can be detected.

To realise the AOC model, three metrics have been proposed: A_{SOT}, A_{PTC} and A_{CTC}. Through the analysis of A_{SOT} and A_{PTC} metrics, whether current OWDs are above the asymptotic OWD line or not can be known. The A_{CTC} metric value is used to judge the OWD's rising strength. Through a comparison with PCT/PDT based algorithms, the merits of the proposed AOC algorithms have been well demonstrated in a number of

typical practical cases. In addition to that, this research has proposed three elaborate parameters: T, K and δ for managing the turning area detection. T is used to remove outliers of OWDs; K serves for the trend detection within a group of OWDs while δ is used for the avail-bw turning area detection.

A fast Avail-bw tool: Pathpair

To validate the proposed AOC model, an avail-bw tool *Pathpair* has been developed in C++. *Pathpair* is a tool based on the two-end model. It was evaluated in the real network - Planetlab environment and was tested over a number of Internet paths with a wide range of network properties. Compared with *Pathload, Spruce and IGI/PTR,* it has been shown that *Pathpair's* estimates were the closest to *Pathload's*[36]. However, the estimation speed has been significantly improved to the same level (1-3 seconds) as the current fastest avail-bw tools, whereas *Pathload* usually requires over 10 seconds per measurement.

7.3 Future Work

In the first part (chapter 2,3,4) of this thesis, a systematic picture of how to develop an automatic traffic classifier have been given as summarised in the previous two sections. However, like other works [MZ05][JEF07], this research was tested in a limited network environment. To apply it in a real large scale Internet environment, more investigations are needed. Similarly, there exist a number of issues that may improve *Pathpair*'s performance and applicability.

[36] 60% of *Pathpair's* avail-bw readings are less than 5% discrepancy with *Pathloads'* and 30% of that are less than 10% different.

7.3.1 Future work in Traffic Classification

Completely overlap issue from unknown traffic classes

As discussed in section 3.7, the KNN-based traffic classifier has the power to recognise different traffic classes. Once a new network traffic category has been recognised, it will be marked according to its pattern position in the produced IP flow profile map. Such information will be stored in the IP flow profile map repository as shown in Fig 4.4. When one type of network traffic comes in to the classifier, the detection result depends on the responded pattern position. If a new position occurs, the detected network traffic will be labelled as a new type of traffic class. Otherwise, it is regarded as the network class that has been registered with the same pattern position as detected. Such a process will repeat successfully except for the case in which a new type (unknown) of network traffic completely overlaps with one registered (known) traffic class. Although this problem can be solved through constructing an alternative map, one issue may exist depending on the degree of overlap. For one unknown network traffic class, if the majority of the responded neurons are completely overlapped[37] with one registered, we may see the detected traffic as the same type as the registered one. If this happens, the proposed automatic classifier will have failed to detect this unknown network traffic class.

In future, more studies are required to address this issue. For example, there are at least two solutions that can be considered. The first solution is to run two or more classification processes in parallel so as to produce a series of IP flow maps. Those processes are executed with a different set of context inputs from one to another. As a result, multiple IP flow maps will be produced. If some traffic classes exhibit a very high degree of complete overlap under one particular input set, the overlaps may degrade with another set of inputs. This solution may be very effective, but clearly it introduces considerable computation

[37] It is different with the case where only one or two neurons overlap as shown in Fig 4.2.

overhead. The other solution is to integrate the proposed classification scheme with other detection methods eg using ontology, as discussed in section 3.3.

Traffic pattern reading in an IP flow profile

This thesis has not gone into details to address how to read different traffic patterns in an IP flow profile. When traffic patterns are well separated (e.g., with large distance between them), recognition of the different traffic class can be easily realised. For instance, in Figure 7.2, there are only two traffic patterns located in different area. In such a case, recognition can be realised by the associated rectangle shape, which is drawn according to the related first and last neuron positions, like (12,0) and (20,15) in Realplayer's case as shown in Fig 7.2. But, when the number of traffic classes grows, it is maybe not such a good option to mark traffic patterns using a rectangle shape; an alternative shape, such as a circle shape[38] may be better. When partial overlap happens, as discussed in section 4.5 and 4.6, one possible reading of different traffic patterns is through the LDA's projection line according to the separation point. In future, more research needs to be carried out on how to mark the traffic patterns in an IP flow profile.

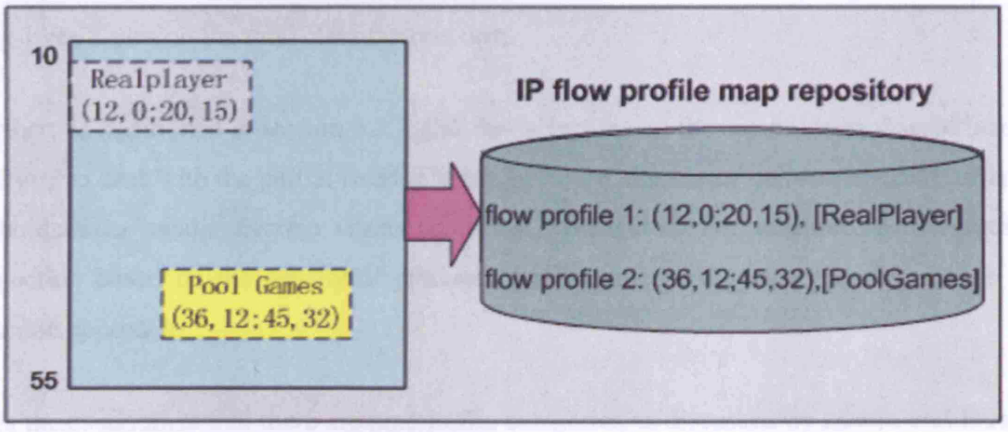


Figure 7.2: The reading of traffic patterns in an IP flow profile map

[38] As discussed in section 4.3.1.

More validations

In future, more tests are required to validate the proposed classifier as a practical tool, particularly tests in a wider scale IP network environment to allow practical performance evaluations. Specifically, these tests could answer the following equations.

- *How many network traffic classes are partially overlapped?*
- *How many network traffic classes are completely overlapped?*
- *Is the use of alternative maps effective in dealing with the complete overlap issue?*
- *Are multiple IP flow maps needed to deal with the complete overlap issue?*
- *Will the classification computation cost be bearable for network operators?*

Aside from the above tests, it will be also interesting to know the impact of classifying long-lived flows on the total classification cost.

Further, as mentioned in section 4.3.2, this thesis focuses on the use of linear discriminant analysis to deal with the partial overlap issue. In future, the use of non-linear discriminant techniques to handle overlap issues could be investigated. For example, a non-linear projection based on the density of produced traffic patterns (as shown in Fig 5.6) is a possible approach.

This thesis assumes that there are nine traffic categories as discussed by Moore and Duev [MZ05](see Section 2.2). Under such an assumption a specific application can be easily extracted according to its pattern position in the IFPM. In future, this may become difficult as the traffic class number grows. To deal with traffic class growth, more investigation is required to improve traffic classification through, for example, the

construction of multiple layer KNN maps, an approach which has been discussed in [CSO96].

7.3.2 Future work in Avail-bw Estimation

Pathpair currently is still a beta version. More efforts are required to improve the coding of *Pathpair* to enable practical implementation. Moreover, more studies need to be carried out on how to deal with the issue that arises when the decreasing part of OWDs, caused by IC, is larger than 50% of a packet train. Furthermore, how to estimate avail-bw of a high speed network path (i.e. the path's bottleneck>1Gbps) still needs more investigation. In addition, more research is needed on how to use the multiple-regression model to improve the avail-bw calculation as discussed in [MBG00][MBG02][CMC05]. Furthermore, due to the testbed limitations, the effect of cross traffic on the *Pathpair's* accuracy has not been evaluated in this thesis. In the latest study [JPW07], it has been shown how cross traffic affects the estimation results of *Pathload* and *Spruce*:

> *"In a 30 minute-long experiment in a traffic scenario using Internet-like bursty cross traffic, only 57% of PATHLOAD estimates and 41% of SPRUCE estimates fall within the same window of accuracy."*

Such testing outcomes comply with the study from [SP04], in which Schormans and Pitts state that there are mainly two kinds of error that may happen to an active probing tool.

> *"... there are two main sources of measurement error. The rate and size of active probes contribute to additional load - the packet probing overhead - and hence greater delays and losses. The rate of the probes and the measurement period determine the number of samples, and hence the sampling error. Both sources of error are worse when the traffic patterns are* ***bursty.*** *"*

Hence, more investigation about the impact of cross traffic on *Pathpair* is required in future.

Last but no least, as mentioned in section 6.5, Pathpair will continue to inject additional probing traffic into the network until it receives the stop signal. In practice, if Pathpair was widely deployed, for inter-domain available bandwidth measurement by a large number of ISP's, this may lead to the network being loaded with a large amount of additional probing traffic; thereby reducing the available bandwidth for data. This is clearly a practical implementation issue that should be further studied in the future, like the introduction of time gap between each packet train.

Reference

[AAP06] D. Antoniades, M. Athanatos, A. Papadogiannakis, E.P. Markatos, and C. Dovrolis. "Available bandwidth measurement as simple as running wget", In Proceedings of the Passive and Active Measurement Conference (PAM2006), March 2006, Page(s): 61-70.

[AM07] Auld, T. Moore, A. W. Gull, S. F. "Bayesian Neural Networks for Internet Traffic Classification", Neural Networks, IEEE Transactions, Publication Date: Jan. 2007, Volume: 18, Issue: 1, On page(s): 223-239, ISSN: 1045-9227.

[BAS06] BASET, S. A., AND SCHULZRINNE, H., "An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol", In Proceedings of the INFOCOM '06, April 2006 Page(s):1 – 11.

[BHH04] S. Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, D. McGuinness, P. Patel-Schneider and L. Stein. "OWL Web Ontology Language Reference", W3C Recommendation 10 February 2004, available at http://www.w3.org/TR/owl-ref.

[BHL01] T. Berners-Lee, J. Hendler, and O. Lassila, "The Semantic Web", Scientific American, May. 2001, Page(s): 34–43.

[BMMR07] Dario Bonfiglio. Marco Mellia. Michela Meo. Dario Rossi. Paolo Tofanelli, "Revealing Skype Traffic- when randomness plays with you", ACM SIGCOMM Computer Communication Review , Volume 37 , Issue 4 (October 2007),Page(s): 37 - 48 , ISSN:0146-4833.

[BTA06] L.Bernaille, R. Teuxeira, I. Akodkenou, A. Soule, K. Salamatian, "Traffic Classification on the Fly", ACM SIGCOMM Computer Communication Review, vol. 36, no. 2, April 2006,New Year, USA, Page(s):23-26.

[CAS03] Silvana Castano, Alfio Ferrara, Stefano Montanelli, Elena Pagani, Gian Paolo Rossi, "Ontology-Addressable Contents in P2P Networks", Proc. 1st Workshop on Semantics in Peer-to-Peer and Grid Computing (SemPGRID'03), Budapest, May 20 2003, Page(s): 55-68.

[CB97] M.E. Crovella and A. Bestavros, "Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes," IEEE/ACM Trans. Networking, vol. 5, no. 6, 1997, Page(s): 835-846.

[CC96a] Robert L. Carter and Mark E. Crovella", Dynamic Server Selection using Bandwidth Probing in Wide-Area Networks", Technical Report BU-CS-96-007, Boston University, 1996.

[CC96b] Robert L. Carter and Mark E. Crovella, "Measuring Bottleneck Link Speed in Packet- Switched Networks", Technical Report BU-CS-96-006, Boston University, 1996.

[CIS05] CISCO white paper, 2005, "DiffServ the scalable end-to-end quality of service model", http://www.cisco.com/application/pdf/en/us/guest/tech/tk766/c1550/ccmigration_09186a00800a3e2f.pdf

[CLL98] Hao Che, San Qi Li, and Arthur Lin, "Adaptive resource management for flow-based IP/ ATM hybrid switching systems". IEEE/ACM Transactions on Networking, 6(5), October 1998, Page(s):544--557

[CMC05] Alexander Chobanyan, Matt Mutka, Zhiwei Cen, Ning Xi, "One Way Delay Trend Detection for Available Bandwidth Measurement", Proceedings of 2005 IEEE GLOBECOM Conference, 2005.

[CSA00] N. Cardwell, S. Savage, T. Anderson, "Modeling TCP Latency," IEEE/INFOCOM2000, Tel-Aviv, Israel, March 2000, Page(s): 1742-1751.

[CSO96] Chen, H., Schuffels, C., and Orwig, R. (1996)," Internet Categorization and Search: A Self-Organizing Approach". Journal of Visual Communication and Image Representation, 7(1), Page(s):88-102.

[DAY90] Dayhoff, J., E., 1990, "Neural Network Architectures", Van Nostrand Reihold, New York, ISBN: 0-442-20744-1.

[DBL03] Parish, D.J., Bharadia, K., Larkum, A., Phillips, I.W. and Oliver, M., "Using packet size distributions to identify real-time networked applications", IEE Proc. Commun. , 1504, August 2003, ISSN 1350-2425, Page(s): 221-227.

[DEY01] Dey, A. K., "Understanding and using context", Journal of Personal and Ubiquitous Computing", Volume 5 (1), 2001, Page(s): 4-7.

[DIJ06] Dijon, France, "The impact of sample reduction on PCA-based feature extraction for supervised learning, Proceedings of the 2006 ACM symposium on Applied computing", ISBN:1-59593-108-2, Page(s): 553 - 558, 2006.

[DOW01a] A. B. Downey. "Evidence for long-tailed distributions in the internet", In Proceedings of ACM SIGCOMM Internet Measurement Workshop, 2001, Page(s):229-241.

[DOW01b] Allen B. Downey, "The structural cause of file size distributions", IEEE MASCOTS01, 2001, Page(s): 328-329.

[DRM01] C. Dovrolis, P. Ramanathan, and D. Moore, "What do Packet Dispersion Techniques Measure?", Proc. of IEEE INFOCOM, Volume 2, 22-26 April 2001, Page(s):905 - 914.

[DRM04] C. Dovrolis, P. Ramanathan, and D. Moore, "Packet Dispersion Techniques and Capacity Estimation", In IEEE/ACM Transactions on Networking, December 2004.

[DWF03] Christian Dewes, Arne Wichmann, and Anja Feldmann, 2003, "An analysis of internet chat systems", In IMC '03: Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement, Page(s): 51-64.

[EAM06] Effrey Erman, Martin Arlitt, Anirban Mahanti, "Traffic Classification Using Clustering Algorithms", SIGCOMM, Workshops September 11-15, 2006, Pisa, Italy. Page(s): 281 - 286.

[EBET03] Sven Ubik, Antonin Kral, "End-to-end Bandwidth Estimation Tools", November 28, 2003, CESNET Technical Report.

[ED01] Brian Everitt, Graham Dunn, 2001, 2nd Edition, "Applied Multivariate Data Analysis", Published by Hodder and Stoughton, ISBN-10 0340741228.

[EKE06] Svante Ekelin, Martin Nilsson, Erik Hartikainen , Andreas Johnsson, Jan-Erik Mings (Ericsson), Bob Melander, Mats Bjorkman, "Real-time Measurement of End-to-End Available Bandwidth Using Kalman Filtering", In proceedings to the Network Operations and Management Symposium, Vancouver, Canada, April, 2006, Page(s):73 – 84.

[EP07] EU-IST project 507134, 2004-2007, Ambient Networks, http://www. ambient-networks.org.

[FF96] Fall, K. and S. Floyd, "Simulation-based Comparisons of Tahoe, Reno and SACK TCP", Computer Communication Review, July 1996. ftp://ftp.ee.lbl.gov/papers /sacks.ps.Z.

[FR00] Freedman, J., "The What, Who, Where, When, Why and How of Context-Awareness", GVU Technical Report, CHI Workshop, work shop 11, 2000.

[FRE01] S.B. Fredj, T. Bonald, A. Proutiere, G. Regnie, and J.W. Roberts, "Statistical bandwidth sharing: A study of congestion at flow level," Proc. ACM SIGCOMM "01, , San Diego, California, U.S.A. Aug. 2001, Page(s):27-31.

[GAL06] Galis, A. L., C K. Jean, et al. "Service-aware Overlay Adaptation in Ambient Networks", International Multi-Conference on Computing in the Global Information Technology (ICCGI) 2006, Page(s): 21 – 29.

[GRU93a] T.R. Gruber. "Towards Principles for the Design of Ontologies Used for Knowledge Sharing", In Roberto Poli Nicola Guarino, editor, International Workshop on Formal Ontology, Padova, Italy, 1993. Page(s): 907–928.

[GRU03b] T. Gruber. "A Translation Approach to Portable Ontology Specification," Knowledge Acquisition, 5(2), 1993, Page(s): 199–220.

[HAG95] Hagiwara, M., 1995, "Self-Organizing Concepts Maps, 22-25 Oct 1995, Systems, Man and Cybernetics", Intelligent Systems for the 21st Century., IEEE International Conference on, Volume: 1,On vol.1, Meeting Date: 22/10/1995 - 10/25/1995, Location: Vancouver, BC, Canada, Page(s): 447-451, ISBN: 0-7803-2559-1.

[HS03] Ningning Hu, Peter Steenkiste, "Evaluation and Characterization of Available Bandwidth Probing Techniques", In the IEEE JSAC Special Issue in Internet and WWW Measurement, Mapping, and Modeling, Aug. 2003, Volume 21, Issue 6, Page(s):879 – 894.

[HSS05] P.,Haffner, S., Sen, O., Spatscheck, D.,Wang , "ACAS:automated construction of application signatures" , Proceeding of the 2005 ACM SIGCOMM workshop on Mining network data, Year of Publication:2005 , ISBN:1-59593-026-4, Page(s): 197 - 202,

[IANA] IANA, Internet Assigned Numbers Authority (IANA), http://www.iana.org /assignments/port-numbers.

[JAC88] V. Jacobson, "Congestion avoidance and control", In Proceedings ACM SIGCOMM88, 1988. Page(s): 314-329.

[JAC90a] V. Jacobson, "Modified TCP congestion avoidance algorithm," message to end2end-interest mailing list, Apr. 1990, URL ftp://ftp.ee.lbl.gov/email/ vanj.90apr30.txt.

[JAC90b] V. Jacobson, "Berkeley TCP evolution from 4.3-tahoe to 4.3-reno," in Proc.18th Internet Engineering Task Force, Vancouver, Aug. 1990.

[JAC92] V. Jacobson, R. Braden, and D. Borman, "TCP extensions for high performance," RFC 1323, May 1992.

[JAC97] V. Jacobson. "*Pathchar* - a tool to infer characteristics of Internet paths", 1997. Presented as April ", 97 MSRI talk, ftp://ee.lbl.gov/pathchar.tar.Z.

[JAM85] James, M. "Classification algorithms", London, Collins Professional and Technical, 1985, ISBN-10: 0471847992.

[JD02] M.,Jain, C. Dovrolis, "*Pathload*: A Measurement Tool for End-to-end Available Bandwidth", In Proceedings of the 3rd Passive and Active Measurements Workshop, March 2002, Fort Collins CO.

[JD03] M. Jain and C. Dovrolis, Aug. 2003, "End-to-end available bandwidth: measurement methodology, dynamics, and relation with TCP throughput," Networking, IEEE/ACM Transactions on, Volume 11, Issue 4, Page(s): 537 – 549.

[JEF07] Erman, Jeffrey; Mahanti, Anirban; Arlitt, Martin; Cohen, Ira; Williamson, Carey , "Offline/realtime traffic classification using semi-supervised learning" Volume 64 , Issue 9-12 (October 2007) , Page(s): 1194-1213, ISSN:0166-5316.

[JMSW07] Hongbo Jiang, Andrew W. Moore, Zihui Ge, Shudong Jin and Jia Wang "Lightweight Application Classification for Network Management" in the Proceedings of the SIGCOMM Workshop on Internet Network Management 2007: The Five-Nines Workshop, August, 2007, Kyoto, Japan.

[JOA01] Joaquim Marques de, "Pattern Recognition", Concepts, Methods and Applications", Springer Verlag. Year 2001, ISBN-10: 3540422978.

[JPW07] Sommers, Joel; Barford, Paul; Willinger, Walter, "Laboratory-based Calibration of Available Bandwidth Estimation Tools", In Elsevier Microprocessors and Microsystems Journal, 31(4), 2007.

[KES91] Srinivasan Keshav, "A Control-Theoretic Approach to Flow Control", In proceeding of SIGCOMM 1991, ACM SIGCOMM,1991, Page(s): 188 – 201.

[JM04] Jawaheer, G.; McCann, J.; 2004, "Building a self-adaptive content distribution network", Database and Expert Systems Applications,. Proceedings 15th International Workshop on 30 Aug.-3 Sept. 2004, Page(s):710 – 713.

[JNG04] Yanxia Jia, Ioanis Nikolaidis , Pawel Gburzynski, "On the effectiveness of alternative paths in QoS routing: Research Articles", International Journal of Communication Systems, v.17 n.1, February 2004, Page(s):1-26.

[KLL06] Eun-Jung Ko, Hyung-Jik Lee, Jeon-Woo Lee, 20-22 Feb. 2006, "Ontology-Based Context-Aware Service Engine for U-HealthCare", Advanced Communication Technology, 2006. ICACT 2006. The 8th International Conference, Page(s): 632- 637.

[KMF04a] Karagiannis, T.; Molle, M.; Faloutsos, M.; Sept.-Oct. 2004, "Long-range dependence ten years of Internet traffic modeling", Internet Computing, IEEE, Volume 8, Issue 5, Page(s):57 – 64.

[KMF04b] T. Karagiannis, M. Molle, and M. Faloutsos., "A Nonstationary Poisson View of Internet Traffic," Proc. IEEE Infocom, IEEE CS Press, 2004, Volume 3, Page(s): 1558 – 1569.

[KOH01] T. Kohonen, Third Extended Edition, "Self-Organizing Maps", Springer Series in Information Sciences, Vol. 30, Springer, Berlin, Heidelberg, New York, 2001. 501 Page(s): ISBN 3-540-67921-9, ISSN 0720-678X.

[KOH88] T. Kohonen, "Self-Organization and Associative Memory". Berlin: Springer-Verlag, August 1988. ISBN: 0387183140.

[KPF05] T. Karagiannis, K. Papagiannaki, and M. Faloutsos, "BLINC: Multilevel Traffic Classification in the Dark". In ACM Sigcomm, Philadelphia, PA, August, 2005. Page(s): 229 – 240.

[LAI05a] Lai, Z., Sokol, J., Eckert, K., P., Galis, A., Lewis, R., Todd, C., "Programmable Network Context Adaptation for Content Delivery Services"- IFIP TC6 Conference,- NetCon'05 –"Network Control and Engineering for QoS, Security and Mobility' Conference; Lannion, France 14-18 November 2005; http://www.netcon05.org /comm.php. Page(s): 243-255.

[LAI05b] Lai, Z., Lewis, R., Galis, A., Todd, C., "Towards Network-Aware Content Delivery Framework"– IEEE International Conference on Software, Telecommunications and Computer Networks 2005, Spit, Marina Frapa, Croatia, 15-17 September 2005, ISBN 953-6114-78-X; http://www.fesb.hr/SoftCOM/2005/.

[LAI05c] Zhaohong, LAI., Galis, A, Lewis, R., Todd, C., "The adaptive Optimal Route Service design for Content Delivery Networks", 9th London Communication Symposium, 8-9 Sept 2005, London, ISBN: 0-9538863-5-2; http://www.ee.ucl.ac.uk/lcs.

[LAI07] Z. Lai, A., Galis, M., Rio, C., Todd, "Towards Automatic Traffic Classification," *ICNS* p.19, International Conference on Networking and Services (ICNS '07), June 2007. Page(s): 19-29, Year of Publication: 2007, ISBN:0-7695-2858-9.

[LB00] Kevin Lai and Mary Baker, "Measuring Link Bandwidths Using a Deterministic Model of Packet Delay", Proceedings of ACM SIGCOMM 2000, August 2000, Page(s):283 – 294.

[LB01] K. Lai and M. Baker. "Nettimer: A tool for measuring bottleneck link bandwidth", Proc. of the USENIX Symp. On Internet Technologies and Systems, San Francisco, California March 2001, Page(s): 11 – 21.

[LC01] Youngseok Lee and Yanghee Choi, "An Adaptive Flow-Level Load Control Scheme for Multipath Forwarding", Lecture Notes In Computer Science; Vol. 2093, Proceedings of the First International Conference on Networking-Part 1 , 2001, Page(s): 771 - 779 , ISBN:3-540-42302-8.

[LEL94] W.E. Leland et al., "On the Self-Similar Nature of Ethernet Traffic" IEEE/ACM Trans. Networking, vol. 2, no. 1, 1994, Page(s):1-15.

[LJ06] Gao Lisha; Luo Junzhou, "Performance Analysis of a P2P-Based VoIP Software", Telecommunications, 2006. AICT-ICIW '06. International Conference on Internet and Web Applications and Services/Advanced International Conference on, 19-25 Feb. 2006 Digital Object Identifier 10.1109/AICT-CIW.2006.Page(s):11- 19.

[LO92] J. Lampinen and E. Oja,"Clustering Properties of Hierarchical Self-Organizing Maps", Journal of Mathematical. Imaging and Vision 2, 1992, Page(s):261-272.

[LSM91] X., lin, D.,Soergel, G.,Marchionini, "A self-organizing semantic map for information retrieva", Proceedings of the 14th annual international ACM SIGIR conference on Research and development in information retrieval, Chicago, Illinois, United States , ,Year of Publication:1991 , Page(s): 262 – 269, ISBN:0-89791-448-1.

[MBG00] B. Melander, M. Bjorkman, and P. Gunningberg, "A new end-to-end probing and analysis method for estimating bandwidth bottlenecks", in Proc. IEEE Globecomm "00, San Francisco, USA, November 2000, Volume 1, Page(s): 415 – 420.

[MBG02] Bob Melander, Mats Bjorkman, and Per Gunningberg. "Regression-Based Available Bandwidth Measurements", Proceedings of the 2002 International Symposium on Performance Evaluation of Computer and Telecommunications Systems, San Diego, CA, USA, July 2002.

[MBG02b] Bob Melander, Mats Bjorrkman and Per Gunningberg, "First-Come-First-Served Packet Dispersion and Implications for TCP". Proceedings of IEEE GLOBECOM-02, Taipei, Taiwan, November 2002. Volume 3, Issue , 17-21 , Page(s): 2170 – 2174.

[MC06] Fivos Constantinou and Panayiotis Mavrommatis, 2006, "Identifying Known and Unknown P2P Traffic". To appear in the proceedings of The 5th IEEE International Symposium on Network Computing and Applications (IEEE NCA06), Page 93-102.

[MG00] I. Matta and L. Guo, "Differentiated Predictive Fair Service for TCP Flows," In Proceedings of ICNP' 2000, 14-17 Nov. 2000, Page(s):49 – 58.

[MG04] A., Malhi and R. X. Gao, "PCA-based Feature Selection Scheme for Machine Defect Classification," IEEE Transactions on Instrumentation and Measurement, Vol. 53, No. 6, 2004, Page(s): 1517-1525.

[MHL04] A. McGregor, M. Hall, P. Lorier, J. Brunskill, "Flow Clustering Using Machine Learning Techniques", Passive & Active Measurement Workshop 2004 (PAM 2004), France, April 19-20, 2004.

[MIK05] Mika Ilvesmaki, "On Traffic Classification and its applications in the Internet", Ph.D thesis, Helsinki University of Technology, Networking laboratory, 2005.

[MIK99] Mika, S.; Ratsch, G.; Weston, J.; Scholkopf, B.; Mullers, K.R.; "Fisher discriminant analysis with kernels" , Neural Networks for Signal Processing IX, 1999. Proceedings of the 1999 IEEE Signal Processing Society Workshop , 23-25 Aug. 1999, Page(s):41 - 48.

[MOL04] Molina, B., et al, 2004, "A closer look at a content delivery network implementation", Electro technical Conference, MELECON 2004. Proceedings of the 12th IEEE Mediterranean, Volume 2, 12-15 May 2004, Page(s): 685 – 688.

[MP05] Andrew W. Moore and Konstantina Papagiannaki, "Toward the Accurate Identification of Network Applications", in the Proceedings of Sixth Passive and Active Measurement Workshop (PAM 2005), March/April 2005, Boston, MA, Page(s): 41-54.

[MZ05] Andrew W. Moore and Denis Zuev, "Internet Traffic Classification Using Bayesian Analysis Techniques" in the Proceedings of the ACM SIGMETRICS June 2005, Banff, Canada, Page(s): 51-64.

[NAI02] Naing, M.-M, Ee-Peng Lim, Dion Goh Hoe-Lian, "Ontology-based web annotation framework for hyperlink structures", Web Information Systems Engineering (Workshops), 2002. Proceedings of the Third International Conference on, Page(s): 184- 193.

[NF06] Cisco IOS. Netflow white papers, 2006,http://www.cisco.com/en/US/products /ps6601/prod white papers list.html.

[NIE04] Norbert Niebert, Andreas Schieder, Henrik Abramowicz, Christian Prehofer, Holger Karl, "Ambient Networks: An Architecture for Communication Networks Beyond 3G", IEEE Wireless Communications, IEEE Wireless Communications, 11(2) April 2004, Page(s): 14-22.

[NLM96] P. Newman, Tom Lyon and G. Minshall, "Flow Labeled IP: Connectionless ATM Under IP", in Proceedings of INFOCOM'96, SanFrancisco, April 1996.

[OD01] Oxford dictionary, http://www.askoxford.com/conciseoed/application?view=uk.

[OGT06] Roel Ocampo , Alex Galis , Chris Todd ,Hermann De Meer, "Towards Context-Based Flow Classification", International Conference on Autonomic and Autonomous Systems (IEEE ICAS'06), Silicon Valley, USA, July 19-21, 2006, Page(s):47-53.

[OR93] J. Oikarinen and D. Reed, "Internet Relay Chat Protocol, RFC 1495," 1993.

[ORR] T. Oetiker and D. R and. "Multi Router Traffic Grapher". http://people.ee.ethz.ch /~oetiker/webtools/mrtg.

[PAX94] V. Paxson, "Empirically-Derived Analytic Models of Wide-Area TCP Connections," IEEE/ACM Transactions on Networking, 2(4), August, 1994, Page(s): 316-336.

[PAX95] Paxson, V.; Floyd, S.; June 1995, "Wide area traffic: the failure of Poisson modeling", Networking, IEEE/ACM Transactions on, Volume 3, Issue 3, Page(s):226 - 244.

[PD04] R. Prasad and C. Dovrolis, "Effects of Interrupt Coalescence on Network Measurements", with. In proceedings of Passive and Active Measurement (PAM) Workshop, April 2004, France.

[PDM03] R, Prasad. Dovrolis, C., Murray, M., Claffy, K.,"Bandwidth estimation: metrics, measurement techniques, and tools, Network", IEEE, Publication Date: Nov.-Dec. 2003, Volume: 17, Issue: 6, Page(s): 27- 35,ISSN: 0890-8044.

[PEC04] Pechenizkiy, M., Tsymbal, A., Puuronen, S., "PCA-based feature transformation for classification: issues in medical diagnostics", Computer-Based Medical Systems, 2004-CBMS, Publication Date: 24-25 June 2004, Page(s): 535- 540, ISSN: 1063-7125.

[PLANET] Planetlab, http://www.planet-lab.org/.

[RCR00] V. J. Ribeiro, M. Coates, R. H. Riedi, S. Sarvotham, and R. G. Baraniuk,"Multifractal cross traffic estimation". In Proc. of ITC specialist seminar on IP traffic Measurement, September 2000.

[RFC1633] IntServ, RFC, 1994, http://rfc.net/rfc1633.html.

[RFC2001] RFC2001, "TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms ", http://www.faqs.org/rfcs/rfc2001.html.

[RFC2474] Definition of the Differentiated Services Field (DS Field), in the IPv4 and IPv6 Headers, 1998, http://www.ietf.org/rfc/rfc2474.txt.

[RFC2475] DiffServ, RFC, 1998, http://rfc.net/rfc2475.html.

[RFC768] J.Postel 28 August 1980, RFC768 , "User Datagram Protocol", http://www.ietf.org/ rfc/rfc768.txt.

[RFC791] Internet Protocol, 1981, http://www.ietf.org/rfc/rfc0791.txt.

[RFC793] RFC 793, September 1981, "TRANSMISSION CONTROL PROTOCOL", http://www.faqs.org/rfcs/rfc793.html.

[RRB03] V. J. Ribeiro, R. H. Riedi, R. G. Baraniuk, J. Navratil, and L. Cottrell. "*pathChirp*: Efficient Available Bandwidth Estimation for Network Paths". In Passive and Active Measurement Workshop, 2003.

[RSS04] M. Roughan, S. Sen, O. Spatscheck, and N. Duffield. "Class-of-Service Mapping for QoS: A Statistical Signature-based Approach to IP Traffic Classification". In ACM/SIGCOMM IMC, November 2004, Page(s): 135-148.

[SAV99] S. Savage. "Sting: a TCP-based Network Measurement Tool". In Proceedings of USENIX Symposium on Internet Technologies and Systems, 1999, Page(s):71-79.

[SBW05] Joel Sommers, Paul Barford, Walter Willinger, "A Proposed Framework for Calibration of Available Bandwidth Estimation Tools", *iscc*, 11th IEEE Symposium on Computers and Communications (ISCC'06), 2006, Page(s): 709-718.

[SCH01] Schmidt, A., K. van Laerhoven, K., "How to Build Smart Appliances?", IEEE Personal Communications 8(4), August 2001, Page(s): 66 – 71.

[SCH99] Schmidt, A., Beigl, M., Gellersen, H.W., "There is more to context than location", Computers & Graphics Journal, Elsevier, Volume 23, No. 6, December 1999, Page(s): 893-901.

[SGB00] R., Stevens, C.A. Goble, and S.,Bechhofer. "Ontology-based Knowledge Representation for Bioinformatics". Briefings in Bioinformatics, 1(4), November 2000, Page(s): 398-416.

[SGG02] S. Saroiu, P. Gummadi and S. Gribble. "*SProbe*: A Fast Technique for Measuring Bottleneck Bandwidth in Uncooperative Environments". In Proceedings of IEEE INFOCOM, 2002, Page(s): 27-38.

[SHJ06] Christoph Schmitz and Andreas Hotho and Robert J., schke and Gerd Stumme. "Content Aggregation on Knowledge Bases using Graph Clustering". In York Sure and John Domingue, editor(s), The Semantic Web: Research and Applications, Springer, Heidelberg, 2006, Page(s): 530-544.

[SKK03] Jacob Strauss, Dina Katabi, and Frans Kaashoek, "A Measurement Study of Available Bandwidth Estimation Tools", The Internet Measurements Conference, Florida, 2003, Page(s): 39-44.

[SMH05] A. Shriram, M. Murray, Y. Hyun, N. Brownlee, A. Broido, M.Fomenkov and K. Claffy, "Comparison of public end-to-end bandwidth estimation tools on high-speed links", in Proc. Passive and Active Measurement workshop (PAM), 2005, Page(s):222-235.

[SP04] J.A., Schormans, J.M. , Pitts, "So you think you can measure IP QoS?" Telecommunications Quality of Services: The Business of Success, 2004, QoS 2004, IEE, On page(s): 151- 155

[SRS97] A. Shaikh, J. Rexford, and K. G. Shin, "Dynamics of Quality-of-Service Routing with Inaccurate Link-State Information," Tech. Rep.CSE-TR-350-97, Dept. of Electrical Engineering and Computer Science, University of Michigan, November, 1997.

[SRS99] A. Shaikh, J. Rexford, and K. G. Shin. "Load-sensitive routing of long-lived IP flows". In Proceedings of ACM SIGCOMM'99, Cambridge, Massachusetts, USA, 1999, Page(s): 215-226.

[SSD02] Mario Schlosser, Michael Sintek, Stefan Decker, Wolfgang Nejdl , "A Scalable and Ontology-Based P2P Infrastructure for Semantic Web Services", IEEE Proceedings of the Second International Conference on Peer-to-Peer Computing, Lingkoping, Sweden, 2002, 5-7/Sept. 2002 Page(s):104 – 111.

[SSW04] S. Sen, O. Spatscheck, and D. Wang, "Accurate, Scalable In-Network Identification of P2P Traffic using Application Signatures," Proceedings of the 13th International World Wide Web Conference, NY, USA, May 2004, Page(s): 512-521.

[STE94] W. Richard Stevens. "TCP/IP Illustrated, Volume 1: the Protocols". Addison-Wesley, 1994, ISBN: 0-201-63346-9.

[SUH06] Kyoungwon Suh, Daniel R. Figueiredo, Jim Kurose, Don Towsley, "Characterizing and detecting relayed traffic: A case study using Skype", IEEE INFOCOM 2006.

[SDZ94] Shenker, S.; Clark, D.D.; Lixia Zhang, "Services or infrastructure: why we need a network service model, Community Networking Integrated Multimedia Services to the Home", 1994., Proceedings of the 1st International Workshop on 13-14 July 1994 Page(s):145 – 149.

[THU03] Thurau,C.,Bauckhage,C,.Sagerer,G., "Combining Self Organizing Maps and Multilayer Perceptions to Learn Bot-Behavior for a Commercial Computer Games", Proc of.Game-on 2003, Page(s):119-123.

[WCS01] Weihua Wang; Chien-Chung Shen, "An adaptive flow classification scheme for data-driven label switching networks", 2001, IEEE International Conference, Volume 8, 11-1, June 2001 Page(s):2613 – 2619.

[WOL03] Hyrdle, Wolfgang, "Applied multivariate statistical analysis", Berlin:Springer-2003. ISBN: 0130925535.

[WS04] Sen, S.; Jia Wang ;"Analyzing peer-to-peer traffic across large networks Networking", IEEE/ACM Transactions on, Volume 12, Issue 2,April 2004,Digital Object Identifier 10.1109/TNET.2004.82627, Page(s):219 – 232.

[WC96] Z.Wang and J. Crowcroft, "QoS routing for Supporting resource reservation", IEEE Journal of Selected Areas of Communications, 14(7):1228--1234, 1996. September.

[WZA06] N. Williams, S. Zander, G. Armitage, "Evaluating Machine Learning Algorithms for Automated Network Application Identification" (pdf) CAIA Technical Report 060410B, April 2006.

[WZA06] N.,Williams, S.,Zander, G.,Armitage, "A preliminary performance comparison of five machine learning algorithms for practical IP traffic flow classification", ACM SIGCOMM Computer Communication Review, Volume 36 , Issue 5 ,(October 2006), ISSN:0146-4833, Page(s): 5 – 16.

[YM01] Yilmaz, S.; Matta, I., "On class-based isolation of UDP, short-lived and long-lived TCP flows Modeling, Analysis and Simulation of Computer and Telecommunication Systems", 2001. Proceedings of Ninth International Symposium on 15-18 Aug. 2001 Page(s): 415 – 422.

[YZX06] Liu Yao, Sui Zhifang, Chen Xuefei, "On Method and Automatic Construction Theory of Domain Ontology Based on Depended Text, Innovative Computing, Information and Control", 2006. ICICIC '06, Volume: 2, Aug. 2006, Page(s): 63 – 66.

[ZAG06] Wenzhe Zhou, Marshall, A., Qiang G., "A novel classification scheme for 802.11 WLAN active attacking traffic patterns", Wireless Communications and Networking Conference, 2006, WCNC 2006. IEEE, Volume: 2, On page(s): 623-628.

[ZM05] Denis Zuev and Andrew W. Moore, "Traffic Classification using a Statistical Approach", in the Proceedings of Sixth Passive and Active Measurement Workshop (PAM 2005), March/April 2005, Boston, MA.

[ZNA05] S. Zander, T.T.T. Nguyen, G. Armitage, "Automated Traffic Classification and Application Identification using Machine Learning", Proceedings of IEEE 30th Conference on Local Computer Networks, Sydney, Australia, 15-17 November 2005, Page(s): 250 – 257.

[ZZP06] Qian Zhao, Yu Zhou Perry, M., "Agreement-aware Semantic Management of Services", Autonomic and Autonomous Systems, 2006. ICAS '06. 2006 International Conference on, 19-21 July 2006.

[Zhao04a] Zenghua Zhao, Yantai Shu, "Multipath traffic distribution in MPLS network", Electrical and Computer Engineering, 2004. Canadian Conference on, Volume 2, 2-5 May 2004 Vol.2, Page(s):661 – 664.

[Zhao04b] Zenghua Zhao; Yanti Shu; Lianfang Zhang; Hongmei Wang; Yang, O.W.W.; "Flow-level multipath load balancing in MPLS network", Communications, 2004 IEEE International Conference on, Volume 2, 20-24 June 2004 , Page(s):1222-1226.

Appendix A: List of publications

The following papers were written or co-written by the author of this thesis during the Ph.D study.

Peer-reviewed conferences, workshops and symposia (Note the papers 1, 2, 4, 7, 10 are related to the thesis)

1. **Z., Lai.**, Galis, A, Lewis, R., C. Todd, -"The adaptive Optimal Route Service design for Content Delivery Networks"- 9th London Communication Symposium, 8-9 Sept 2005, London, ISBN: 0-9538863-5-2; http://www.ee.ucl.ac.uk/lcs.

2. **Z., Lai,** R. A. Galis, , C. Todd, -"Towards Network-Aware Content Delivery Framework"– IEEE International Conference on Software, Telecommunications and Computer Networks 2005, Spit, Marina Frapa, Croatia, 15-17 September 2005, ISBN 953-6114-78-X; www.fesb.hr/SoftCOM/2005/.

3. R., Ocampo, L., Cheng, , **Z. Lai,** , and Galis, A., -"ContextWare Support for Network and Service Composition and Self-Adaptation"-; IEEE MATA 2005 - Mobility Aware Technologies and Applications, - Service Delivery Platforms for Next Generation Networks; Springer ISBN-2 553-01401-5; 17-19 October 2005, Montreal, Canada; www.congresbcu.com/mata2005/;

4. **Z. Lai,** J. Sokol, P.K.Eckert, A. Galis, C, Todd, –"Programmable Network Context Adaptation for Content Delivery Services"- IFIP TC6 Conference,- NetCon'05 –"Network Control and Engineering for QoS, Security and Mobility' Conference; Lannion, France 14-18 November 2005; www.netcon05.org/comm.php, Page(s): 243-255.

5. C., Simon, A.Wagner, P. Kersch, , M. Erdei, T. Katona, B.Benk, , Szabó, R., A.Galis, L., Cheng, , K.,Jean, , **Z., Lai,**- "Peer-to-Peer Management in Ambient Networks"– paper and demonstration at IST Mobile and Wireless Communications Summit, Myconos, 4-8 June 2006, http://mobilesummit2006.org/ms2006/servlet/org.nkpap.visualizer.Main.

6. L. Cheng, R. Ocampo, K. Jean, A. Galis, **Z., Lai,** et al, "Distributed Hash Tables Composition in Ambient Networks: A Synthesis Study" - accepted on 14th July 2006 as a full paper at IEEE DSOM'06, 23-25 October 2006, Dublin, Ireland.

7. **Z., Lai**., A, Galis, C. Todd, -"The Adaptive Overlay Routing of long-lived flows in Ambient Networks"- 10th London Communication Symposium, Sept 2006, London;

8. Roel O, Lawrence C, Kerry J, Alberto P, Alex G, **Z., Lai**, "Towards a Context Monitoring System for Ambient Networks", - 2006 IEEE CHINACOM 2006, 16-19 October 2006, Beijing.

9. B., Mathieu, , M. Song, A. Galis, L. Cheng, , Jean, K., Ocamo, R., **Z., Lai**, M.,Brunner, , M.,Stiemerling,M., Cassini, -"Self-Management of Context-Aware Overlay Ambient Networks" - 10th IFIP/IEEE International Symposium on Integrated Management (IM 2007); 21-25 May 2007, Munich, Germany; http://www.im2007.org.

10. **Z., Lai,** A., Galis, M., Rio, C., Todd, "Towards Automatic Traffic Classification," *ICNS* p.19, International Conference on Networking and Services (ICNS '07), June 2007. Page(s): 19-29, Year of Publication: 2007, ISBN:0-7695-2858-9.

11. B. Mathieu, , A.Galis, , K.Jean , Ocampo, R., **Z., Lai**, , Stiemerling, M., Cano,M., Kampmann, M., Tariq, M., Balos, K., Ahmed, K., Busropan, B., Prins, M.,-"Dynamic Adaptable Overlay Networks for Personalised Service Delivery" - IEEE 7th International Symposium on Communications and Information Technologies - http://www.elec.uow.edu.au/ISCIT2007/ M2NM 2007- First Ambient Networks Workshop on Mobility, Multiaccess, and

Network Management; 16-19 Oct 2007, Sydney, Australia; and http://nicta.com.au/research/projects/ambient_networks/m2nm-2007.

12. B., Mathieu, , M. Song, A. Galis, L. Cheng, , Jean, K., Ocamo, R., **Z., Lai,** , Brunner, M., Stiemerling, M., Cassini, M., Kampmann, M. -"Autonomic Management of Context-Aware Ambient Overlay Networks" - IEEE International Conference on Communications and Networking in China (IEEE CHINACOM 2007), 15-17 August 2007, Shanghai, www.chinacom.org/2007.

Appendix B: KNN Codes

B.1: Kohonen Self-Organising Prototype with Matlab (KNN.m)

```
% Kohonen training program
% The second layer grid structure: MxN matrixes
% The Input Number: E;
% Weight: U[M,N,E];
% training data: Tdata;

clear;
M=25; N=M; E=3;
InitNeighborhoodSize=sqrt(M);
NeighborhoodSize=InitNeighborhoodSize;
CollapseRate=0.05;

% step 1: initialising input, weight data, learning rate
%Tdata=rand(1500,E)*0.8+0.1; % random data for input ranging from 0.1 to 0.9

v1 = rand(6000,E)*0.98+0.01;

Tdata=[v1];

%Tdata=[v1(:,1),v1(:,4);v1(:,1),v1(:,2);v1(:,1),v1(:,3);v1(:,2),v1(:,3);v1(:,2),v1(:,4);v1(:,3)
,v1(:,4);v2];


U=rand([M,N,E]);                    % random data for initial weights

%U(:,:,1)=dlmread('w1.txt',' ');
%U(:,:,2)=dlmread('w2.txt',' ');


alpha=0.1;                          % learning rate
init_alpha=alpha;
Iter=0;
```

```
%kohshow
wx=[];wy=[];

for Iter=1:100
    NeighborhoodSize=InitNeighborhoodSize*(1-0.05)^(Iter-1);
    %alpha=init_alpha^(Iter-1);
    fprintf('NeighborhoodSize=%f\n',NeighborhoodSize);
    if (NeighborhoodSize<=0.6) break; end

     fix(clock)

     for coord_x=1:M
       for coord_y=1:N
           for i=1:M
         x_dist=(i-coord_x)^2;
         for j=1:N
                        xy_dist=x_dist+(j-coord_y)^2;
                        gaussian_matrix(i,j)=exp(-xy_dist/(NeighborhoodSize^2))*alpha;
         end
           end
         sigma{coord_x,coord_y}=gaussian_matrix;
       end
     end

% step 2: to calculate the winner
    for data_no=1:size(Tdata,1);
          dist_kt=0;
          for index=1:E;
            dist_kt=dist_kt+(Tdata(data_no,index)-U(:,:,index)).^2;    % to add all the
vectors together
          end
          [min_data,winner_no]=min(dist_kt(:));
          coord_y = 1+floor((winner_no-1)/M);                          % to get x,y
coordinate
          coord_x = winner_no-(coord_y-1)*M;

          wx=[wx,coord_x];
          wy=[wy,coord_y];

        % step 3: to update the weight
                for index=1:E
                    U(:,:,index)=       U(:,:,index).*(1-sigma{coord_x,coord_y})       +
Tdata(data_no,index)*sigma{coord_x,coord_y};
                end
    end
    fprintf('Iter=%d..',Iter);
    %kohshow
```

end

B.2: KNN Testing Prototype with Matlab (TM.m)

```
clear;

N=80;M=N;
E=3;

U(:,:,1)=dlmread('w80.1.txt');
U(:,:,2)=dlmread('w80.2.txt');
U(:,:,3)=dlmread('w80.3.txt');



v1=dlmread('skype.new.txt');
v2=dlmread('p2p.txt');
v3=dlmread('realplay-1.2.4.txt');
v6=dlmread('ftp.txt');
v7=dlmread('bt.txt');
v8=dlmread('msn.voice.txt');
v9=dlmread('pool.original.txt');
v10=dlmread('mediaplay.txt');
skype=[v1(:,1),v1(:,2),v1(:,5)];
realplay=[v3];
p2p=[v2(:,1),v2(:,2),v2(:,4)];
ftp=[v6(:,1),v6(:,2),v6(:,4)];
bt=[v7(:,1),v7(:,2),v7(:,4)];
msn_voice=[v8(:,3)/68086,v8(:,2),v8(:,4)];
pool=[v9(:,1),v9(:,2),v9(:,4)];
mediaplay=[(v10(:,1)./(v10(:,1)+1)),v10(:,2),v10(:,3)];


data(N,N)=0;
      fix(clock)
 Tdata=[realplay];      [X,Y,data]=SOM(Tdata);      a1=1:N; a2=a1; a3=data(a1,a2);
 xy=[data];
figure(1);      plot(Y,X,'o','MarkerFaceColor', 'm');      axis([1 N 1 N]);      hold on;
realplay_g=[X;Y];
Tdata=[skype];                [X,Y,data]=SOM(Tdata);                figure(1);
plot(Y,X,'*','MarkerFaceColor',          'b','MarkerEdgeColor','c');          xy=xy+data;
skype_g=[X;Y];
```

```
Tdata=[p2p];                [X,Y,data]=SOM(Tdata);                figure(1);
plot(Y,X,'h','MarkerFaceColor', 'g');xy=xy+data; p2p_g=[X;Y];
Tdata=[ftp];                [X,Y,data]=SOM(Tdata);                figure(1);
plot(Y,X,'d','MarkerFaceColor', 'r');xy=xy+data; ftp_g=[X;Y];
Tdata=[pool];               [X,Y,data]=SOM(Tdata);                figure(1);
plot(Y,X,'s','MarkerFaceColor', 'y');xy=xy+data; pool_g=[X;Y];
%Tdata=[mediaplay];         [X,Y,data]=SOM(Tdata);                figure(1);
plot(Y,X,'s','MarkerFaceColor', 'k');xy=xy+data;
    fix(clock)

 h=legend('MULTIMEDIA(Realplayer)','CHAT(skype)','P2P(bt)','BULK(ftp)','GAME
 S(pool)');

%h=legend('Realplayer','Mediaplayer','Pool',3);
%xlabel('1-80 Neurons');
 title('Traffic Classification Results on 80x80 Self-Organising Feature Map'); %
 ('FontSize',16);

%h = legend('cos','sin',2);

grid on;
hold off
```

SOM.m

```
function [X,Y,data] =   SOM(Tdata)

    N=80;M=N;
    E=3;
    data=zeros(N,N);

    X=[];Y=[];
    U(:,:,1)=dlmread('w80.1.txt');
    U(:,:,2)=dlmread('w80.2.txt');
    U(:,:,3)=dlmread('w80.3.txt');
    %data(N,N)=0;
    for data_no=1:size(Tdata,1);
        dist_kt=0;
        for index=1:E;
          dist_kt=dist_kt+(Tdata(data_no,index)-U(:,:,index)).^2;    % to add all the
vectors together
        end
        [min_data,winner_no]=min(dist_kt(:));
```

```
        coord_y = 1+floor((winner_no-1)/M);                    % to get x,y
coordinate
        coord_x = winner_no-(coord_y-1)*M;
    X(data_no)=coord_x;
    Y(data_no)=coord_y;
     data((N+1-coord_x),coord_y)=data((N+1-coord_x),coord_y)+1;   % here as to show
the correct data position, using the corrd_y - Lai 31/Oct
    end
    p=sort([Y;X]');
```

Appendix C: KDE, PCA and LDA Codes

C.1 Kernel Density Estimation Program

```
%
%   Kernel Density Estimation Program
%
%

clear;

%v1=dlmread('size.bbcrp.txt');

v1=dlmread('rp.allsize.200.txt');

%v=[v1(1:2500,:)];
v=v1(:,2);
% step 1: to plot the data with bar according to the frequency data

binwidth=20;
x=0:binwidth:1500;
xh=x+binwidth/2;                % to get the central point
[n]=histc(v,x);          % returns the frequency - n
bar(xh,n/(length(v)*binwidth),1);

% step 2: to draw KDE with the default bandwidth

xi=0:5:1500;
[f,xi,w]=ksdensity(v,xi);    % w refers to the bandwidth of KDE and equal to the default
setting of Matlab, f is the frequency within each bandwidth
line(xi,f);
w

% step 3: to reduce bandwidth to 10,

[f1,xi,w]=ksdensity(v,xi,'width',5);
line(xi,f1,'color','red');
line(v,v.*0,'Linestyle','none','Marker','.','Markersize',14);
```

```
%[f,xi,w]=ksdensity(v,xi,'width',20);
%line(xi,f,'color','yellow');

%[f,xi,w]=ksdensity(v,xi,'width',50);
%line(xi,f,'color','green');
```

C.2 Principal Component Analysis Program

```
% Principle Component Analysis
% Writting by Zhaohong Lai at UCL
% Year 2006-2007

clear;


v1 = dlmread('pool.txt');
%v2=[v1(:,1),v1(:,2)/1500,v1(:,3)/1000000,v1(:,4)/1500,v1(:,5)/1000000];

Tdata=[v1];

M=size(Tdata,2);
N=M;

for i=1:N
      m(1,i)=mean(Tdata(:,i));
      sd(:,i)=(Tdata(:,i)-m(1,i))/std(Tdata(:,i));
end

[R,P]=corrcoef(Tdata);

[A,D]=eigs(R,N);

W=[sd'];

%y1=A(:,1)*W;
%y2=A(:,2)*W;

Y=A'*W;
```

```
for i=1:N
      minabs=abs(min(Y(i,:)));
      Y(i,:)=Y(i,:)+minabs+0.01;
      maxval=max(Y(i,:))+0.02;
      Y(i,:)=Y(i,:)/maxval;
end

 fid = fopen('r3.txt','w');
 fprintf(fid,'%f    %f %f %f\n',Y);
 fclose(fid);
```

C.3 Linear Discriminant Analysis

```
%
%   Linear Discriminant Analysis
%
%
    clear;

    v1=dlmread('skype_g2.txt');
    data1=[v1(2,:);v1(1,:)];
    v2=dlmread('pool_g2.txt');
    data2=[v2(2,:);v2(1,:)];

    fig=figure;
    plot(data1(1,:),data1(2,:),'d','MarkerFaceColor','r'); hold on;
    plot(data2(1,:),data2(2,:),'h','MarkerFaceColor','g');

    %plot(data1(1,:),data1(2,:),'r.');
    %plot(data2(1,:),data2(2,:),'g.');

    grid on;
    axis([1 50 1 50]);

    p1=size(data1,2)/(size(data1,2)+size(data2,2));
    p2=size(data2,2)/(size(data1,2)+size(data2,2));

    m1=mean(data1')'; m2=mean(data2')';
    origin=mean([m1 m2]')';
    %origin=p1*m1+p2*m2;

    plot(origin(1),origin(2),'s','MarkerFaceColor','k');
```

```
    %between-class scatter
    sb=(m1-m2)*(m1-m2)';    % Note,    Sb=(N1*N2/N)*sb where SB to the usual
definition

    %within-class scatter
    [d n]=size(data1);
    data=data1-m1*ones(1,n);
    sw1=(n-1)*cov(data');    % Note: N-1 times cov to add weight for within class,
equivelant to p*cov
    [d n]=size(data2);
    data=data2-m2*ones(1,n);
    sw2=(n-1)*cov(data')';
    sw=sw1+sw2;

    [evec eval]=eigs(sb,sw);
    V=evec; D=eval;

    %[eval,iEvals]=sort(diag(eval));
    %iEvals=flipud(iEvals);
    %evec=evec(:,iEvals);
    %eval=eval(iEvals);
    v=V(:,1);

    %v=v/norm(v);

    d1=[v'*data1]';
    d2=[v'*data2]';
    md=[v'*origin];

    fprintf('md=%f\nmin_d1=%f                                    max_d1=%f\nmin_d2=%f
max_d2=%f\n',md,min(d1),max(d1),min(d2),max(d2));

    True=0;False=0;
    for i=1:size(d1,1)
        if(d1(i)>=md) True=True+1;
        else False=False+1;
        end
    end
    sucR=True/size(d1,1);
    fprintf('Suc=%d False=%d sucR=%f \n',True, False, sucR);

    True=0;False=0;
    for i=1:size(d2,1)
        if(d2(i)<md)
            True=True+1;
        else
```

```
            False=False+1;
            fprintf('i=%d ',i);
        end
end

sucR=True/size(d2,1);

fprintf('\nSuc=%d False=%d sucR=%f ',True, False, sucR);
```

Appendix D: Hidden Hop and Smallest Surplus First Issues

D.1: Hidden Hop Issue

Let u stand for the probing traffic rate, X be the cross-traffic rate and C be the link capacity, then the receiving rate r and the cross traffic crate X can be calculated as follows [MBG00]:

$$r = \frac{u}{u+X} \times C \tag{8.1}$$

$$X = \frac{C-r}{r} \times u \tag{8.2}$$

Fig 8.1 presents an example of the hidden hop issue. In this Figure, there are two links in the path. Let us assume that the capacity of link 1 equals 155Mbps with 3Mbps surplus bandwidth; the capacity of link 2 equals 10Mbps with 7Mbps surplus bandwidth and the incoming traffic equals 10 Mbps. According to Equation 8.1, the receiving traffic rate after link 1 and link 2 will be $\frac{10}{10+152} \times 155 = 9.57$ Mbps and $\frac{9.57}{3+9.57} \times 10 = 7.61$ Mbps. Using one of network capacity estimation tool like *Pathrate*, it can be known that the bottleneck of this path is 10Mbps. According to Equation 8.2, the competing traffic X is: $\frac{10-7.61}{7.61} \times 10 = 3.14$ Mbps. As a result, the estimated the available bandwidth will be (10-3.14)=6.86 Mbps. Such a measurement result clearly is wrong as the available bandwidth is 3Mbps according to Equation 5.3. This problem is called the hidden hop issue [MBG00][MBG02].

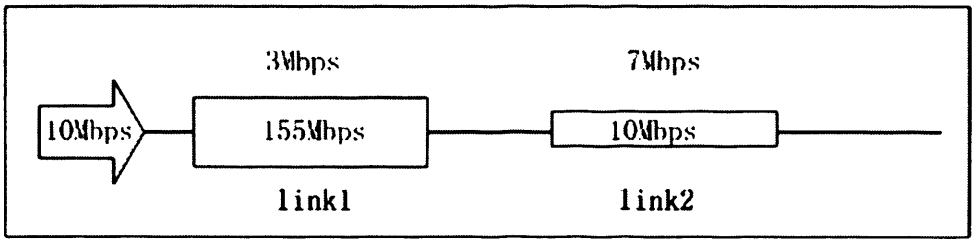


Figure 8.1: The Hidden Hop problem

D.2: Smallest Surplus First Issue

To deal with the hidden hop issue, Melander et al propose the regression-based available bandwidth measurement method. According to Equation 5.13, the ratio between the probing traffic rate and the receiving rate is expressed as follows.

$$\frac{u}{r} = \frac{C-A}{C} + \frac{1}{C}u = (1-\frac{A}{C}) + \frac{1}{C}u = \alpha + \beta u \qquad (u > C - X) \tag{8.3}$$

where A represents avail-bw, $\alpha = 1 - \frac{A}{C}$ and $\beta = \frac{1}{C}$. Hence, the ratio of *u/r* changes linearly according to two parameters α and β. As shown in Equation 8.3, if knowing α and β, both the cross traffic rate *C* and *avail-bw A* can be obtained. Fig 8.2 shows a plot of *u/r* when the path follows the condition of the smallest surplus first (SSF). As it can be seen, the curve is piece-wise linear. This is because there is more than one congestible hop. Hence, the model of the curve actually is segmented linear model. Therefore the multiple regression based calculation can be applied to obtain β [MRG00]. But, the model developed in [MRG00][MRG02] is subject to the SSF condition. Otherwise, the regression based calculation may give incorrect estimates in the non-SFF environment as stated in [MRG02].

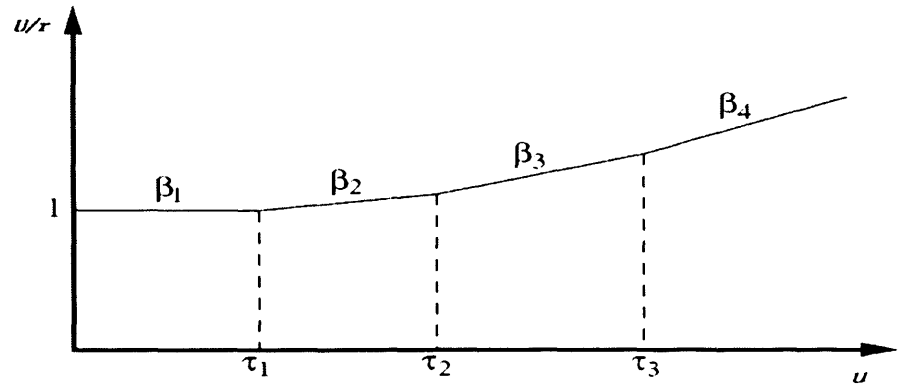


Figure 8.2: The plot of u/r

Appendix E: The route between Kent and Chicago

As traceroute did not work on the Planetlab node in Kent and the Planetlab node in Chicago were not on at the time when author performed the measurements, the route from planetlab1.kent.ac.uk to planetlab1.cs.uchicago.edu can not be obtained directly. But, it can be indirectly inferred through the analysis of the route from UK to Chicago as discussed below.

E.1: The routes of the paths from UK to Chicago

In order to infer the route between Kent and Chicago, we need the route information from some UK nodes to US Chicago. There are two paths that have been tested: the path from planet1.manchester.ac.uk to planetlab1.cs.uchicago.edu and the path from planetlab1.nrl.dcs.qmul.ac.uk to planetlab1.cs.uchicago.edu. The testing results are shown in the following two tables.

Table E.1: The traceroute result from Manchester to planetlab1.cs.uchicago.edu

1	130.88.203.251 (130.88.203.251) 0.413 ms 0.322 ms 0.289 ms
2	gw-uom-kb.its.manchester.ac.uk (130.88.250.42) 0.531 ms 0.390 ms 0.326 ms
3	gw-man-kb.netnw.net.uk (194.66.26.101) 0.299 ms 0.299 ms 0.824 ms
4	gw-man-rh.netnw.net.uk (194.66.27.17) 0.400 ms 0.394 ms 0.318 ms
5	so-0-1-0.warr-sbr1.ja.net (146.97.42.169) 1.156 ms 1.050 ms 1.049 ms
6	so-3-0-0.lond-sbr3.ja.net (146.97.33.18) 6.016 ms 6.004 ms 23.856 ms
7	lond-sbr5.ja.net (146.97.33.2) 6.170 ms 6.006 ms 6.106 ms
8	po1-0.gn2-gw1.ja.net (146.97.35.98) 6.118 ms 10.004 ms 6.088 ms
9	janet.rt1.lon.uk.geant2.net (62.40.124.197) 6.198 ms 8.189 ms 6.195 ms
10	so-2-0-0.rt1.ams.nl.geant2.net (62.40.112.137) 14.344 ms 14.328 ms 17.197 ms
11	so-7-0-0.rt1.nyc.us.geant2.net (62.40.112.134) 97.606 ms 97.677 ms 107.857 ms
12	198.32.11.50 (198.32.11.50) 97.712 ms 103.111 ms 98.674 ms
13	so-0-0-0.0.rtr.wash.net.internet2.edu (64.57.28.11) 127.153 ms 132.891 ms 114.009 ms

14	so-0-2-0.0.rtr.chic.net.internet2.edu (64.57.28.12) 130.672 ms 138.185 ms 130.730 ms
15	mren-chin-ge.abilene.ucaid.edu (198.32.11.98) 147.067 ms 144.347 ms 146.123 ms
16	MREN-IWIRE-10G-local.uchicago.edu (128.135.247.121) 133.650 ms 131.269 ms 131.374 ms

Table E.2: The traceroute result from Queen Mary to planetlab1.cs.uchicago.edu

1	194.36.10.254 (194.36.10.254) 0.199 ms 0.268 ms 0.683 ms
2	ic-gsr.lmn.net.uk (194.83.102.37) 0.493 ms 0.380 ms 0.494 ms
3	so-1-0-0.lond-sbr1.ja.net (146.97.42.61) 2.242 ms 0.768 ms 2.845 ms
4	so-0-0-0.lond-sbr3.ja.net (146.97.33.134) 1.304 ms 2.314 ms 1.857 ms
5	lond-sbr5.ja.net (146.97.33.2) 1.331 ms 47.853 ms 1.437 ms
6	po1-0.gn2-gw1.ja.net (146.97.35.98) 1.963 ms 1.413 ms 1.346 ms
7	janet.rt1.lon.uk.geant2.net (62.40.124.197) 4.468 ms 1.730 ms 18.800 ms
8	so-2-0-0.rt1.ams.nl.geant2.net (62.40.112.137) 10.592 ms 10.149 ms 9.742 ms
9	so-7-0-0.rt1.nyc.us.geant2.net (62.40.112.134) 99.989 ms 93.141 ms 92.920 ms
10	198.32.11.50 (198.32.11.50) 94.564 ms 92.912 ms 100.635 ms
11	so-0-0-0.0.rtr.wash.net.internet2.edu (64.57.28.11) 118.619 ms 132.848 ms 117.508 ms
12	so-0-2-0.0.rtr.chic.net.internet2.edu (64.57.28.12) 125.845 ms 128.957 ms 126.415 ms
13	mren-chin-ge.abilene.ucaid.edu (198.32.11.98) 146.844 ms 140.423 ms 144.039 ms
14	MREN-IWIRE-10G-local.uchicago.edu (128.135.247.121) 135.153 ms 130.920 ms 126.529 ms

According to the above two tables, it can be known that the common part of the route from UK Planetlab nodes to US Chicago consists of 12 nodes as shown in Table E.3 and the relevant RTT is around 130 ms (one is 126.5ms; the other is 131.374ms).

Table E.3: The common part of the route from UK to US Chicago

so-0-1-0.warr-sbr1.ja.net (146.97.42.169) 1.156 ms 1.050 ms 1.049 ms
so-3-0-0.lond-sbr3.ja.net (146.97.33.18) 6.016 ms 6.004 ms 23.856 ms
lond-sbr5.ja.net (146.97.33.2) 6.170 ms 6.006 ms 6.106 ms
po1-0.gn2-gw1.ja.net (146.97.35.98) 6.118 ms 10.004 ms 6.088 ms
janet.rt1.lon.uk.geant2.net (62.40.124.197) 6.198 ms 8.189 ms 6.195 ms
so-2-0-0.rt1.ams.nl.geant2.net (62.40.112.137) 14.344 ms 14.328 ms 17.197 ms
so-7-0-0.rt1.nyc.us.geant2.net (62.40.112.134) 97.606 ms 97.677 ms 107.857 ms
198.32.11.50 (198.32.11.50) 97.712 ms 103.111 ms 98.674 ms
so-0-0-0.0.rtr.wash.net.internet2.edu (64.57.28.11) 127.153 ms 132.891 ms 114.009 ms
so-0-2-0.0.rtr.chic.net.internet2.edu (64.57.28.12) 130.672 ms 138.185 ms 130.730 ms
mren-chin-ge.abilene.ucaid.edu (198.32.11.98) 147.067 ms 144.347 ms 146.123 ms
MREN-IWIRE-10G-local.uchicago.edu (128.135.247.121) 133.650 ms 131.269 ms 131.374 ms

E.1: The route from US to UK Kent

Table E.4 shows the traceroute output of the path from US Connell to UK Kent. This table shows that there are 3 nodes from London so-0-1-0.lond-sbr1.ja.net to Kent's Planetlab node.

Table E.4: The traceroute result from US Connell to UK Kent

1	rhodes1-msfc-vl339.net.cornell.edu (128.84.154.1) 0.308 ms 0.334 ms 0.267 ms
2	core2-6500-vl8.net.cornell.edu (132.236.222.174) 0.463 ms 8.255 ms 1.008 ms
3	nyc1-msfc-dmz2.net.cornell.edu (132.236.222.4) 8.237 ms 8.222 ms 12.833 ms
4	newy-nlr-vl4000.nelr.net (192.35.82.129) 9.490 ms 11.856 ms 9.075 ms
5	216.24.184.86 (216.24.184.86) 8.334 ms 8.359 ms 9.434 ms
6	so-7-0-0.rt1.ams.nl.geant2.net (62.40.112.133) 97.966 ms 99.245 ms 96.353 ms
7	so-4-0-0.rt1.lon.uk.geant2.net (62.40.112.138) 99.869 ms 99.793 ms 141.289 ms
8	po2-0-0.gn2-gw1.ja.net (62.40.124.198) 99.653 ms 101.206 ms 99.606 ms
9	po1-1.lond-sbr5.ja.net (146.97.35.97) 99.956 ms 101.075 ms 108.789 ms
10	lond-sbr3.ja.net (146.97.33.5) 99.719 ms 99.831 ms 99.823 ms
11	so-0-1-0.lond-sbr1.ja.net (146.97.33.133) 102.573 ms 100.273 ms 100.387 ms
12	KentishMAN-K2.site.ja.net (146.97.42.70) 102.310 ms 105.244 ms 107.541 ms
13	uok-c-site.kentman.ac.uk (212.219.171.130) 111.836 ms 102.330 ms 102.296 ms

Hence, by combining table E.3 with table E.4 together, we can know that the route from Kent to Chicago includes the following 14 nodes (Table E.5).

Table E.5: The inferred route from UK Kent to US Chicago

1	drcsa-dmz.kent.ac.uk (129.12.3.66)
2	uok-c-site.kentman.ac.uk (212.219.171.130)
3	KentishMAN-K2.site.ja.net (146.97.42.70)
3	so-1-0-0.lond-sbr1.ja.net (146.97.42.61) 2.242 ms 0.768 ms 2.845 ms
4	so-0-0-0.lond-sbr3.ja.net (146.97.33.134) 1.304 ms 2.314 ms 1.857 ms
5	lond-sbr5.ja.net (146.97.33.2) 1.331 ms 47.853 ms 1.437 ms
6	po1-0.gn2-gw1.ja.net (146.97.35.98) 1.963 ms 1.413 ms 1.346 ms
7	janet.rt1.lon.uk.geant2.net (62.40.124.197) 4.468 ms 1.730 ms 18.800 ms
8	so-2-0-0.rt1.ams.nl.geant2.net (62.40.112.137) 10.592 ms 10.149 ms 9.742 ms
9	so-7-0-0.rt1.nyc.us.geant2.net (62.40.112.134) 99.989 ms 93.141 ms 92.920 ms
10	198.32.11.50 (198.32.11.50) 94.564 ms 92.912 ms 100.635 ms
11	so-0-0-0.0.rtr.wash.net.internet2.edu (64.57.28.11) 118.619 ms 132.848 ms 117.508 ms
12	so-0-2-0.0.rtr.chic.net.internet2.edu (64.57.28.12) 125.845 ms 128.957 ms 126.415 ms
13	mren-chin-ge.abilene.ucaid.edu (198.32.11.98) 146.844 ms 140.423 ms 144.039 ms
14	MREN-IWIRE-10G-local.uchicago.edu (128.135.247.121) 135.153 ms 130.920 ms 126.529 ms